


ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

# "ADAPTACIÓN DEL USO DE ROUTERS CISCO EN PRÁCTICAS DE LABORATORIO A ROUTES OPEN-SOURCE"

Héctor Egea Chueca

Daniel Morató Osés

Pamplona, 1-9-2011

# ÍNDICE

# RESUMEN

En este proyecto se comprueba en qué medida una solución de router open-source, puede sustituir a routers CISCO en un entorno de laboratorio. Se podrá ver algunos de los servicios de los que dispone el sistema Vyatta en su versión gratuita, comprobando que puede realizar muchas de las funciones de los routers CISCO.

Para ello, primero se hará una comparación de las diferentes opciones que existen a la hora de elegir un open-source routing, explicando qué es, cómo surgió, qué servicios ofrece, si está constantemente en desarrollo, etc.

A continuación, se realizará un breve tutorial del sistema Vyatta, en el que se explicará cómo empezar a usar el sistema, enumerando todas las instrucciones que hagan falta. Además, se comentarán todos los servicios, los que se usarán en las prácticas y los que no, explicando en qué consisten y cómo se configurarían con este software.

Por último, se realizarán las prácticas de la asignatura Laboratorio de Programación de Redes (LPR), analizando si se pueden sustituir los routers Cisco por el sistema Vyatta.

# 1.- INTRODUCCIÓN

## 1.1. Objeto del PFC

Hoy en día, la mayoría de las empresas, ya sean grandes o pequeñas, están informatizadas de tal manera que cada trabajador posee un ordenador propio o compartido en el que realiza sus tareas.
Además para que todo funcione correctamente y todos los elementos interrelacionen entre sí, es necesario mencionar el término de red, en concreto Red de Área Local (LAN).

El objetivo de estas redes es conseguir la optimización del uso de recursos hardware (impresoras, escáner, espacio en disco duro, etc.) y software (base de datos, correo electrónico, ficheros, antivirus, etc.), compartiendo aquellos que resulten más costosos, reduciendo la necesidad de espacio físico e incrementando la capacidad de comunicación entre los miembros de la empresa.

Un router o enrutador es un dispositivo que asegura el enrutamiento de paquetes entre redes, o bien determina la ruta exacta que debería tomar el paquete de datos que intercambiamos. En definitiva, un router se utiliza para conectar múltiples redes. En el caso de una empresa, conectar el mundo exterior con algunos de los ordenadores de la empresa, proteger la información de las amenazas de seguridad, e incluso decidir que ordenadores tienen prioridad sobre otros.

En función del negocio y de las conexiones de red, hay routers que incluyen diferentes capacidades. Pueden incluir funciones como la de cortafuegos, red privada virtual (VPN), red telefónica IP, etc.

Por supuesto, el precio de estos dispositivos, que puede variar entre unas cantidades elevadas, hace que unas empresas se decanten por uno u otros en función de su presupuesto. En la actualidad la mayoría de las empresas, poseen routers de la marca Cisco Systems. Ésta marca se caracteriza por desarrollar no sólo el hardware de sus equipos sino su propio software de configuración y gestión de los mismos. Dicho software es conocido como IOS de código actualmente cerrado y totalmente propietario.

Para las grandes empresas y multinacionales, el gasto de este router no supone gran esfuerzo económico, pero para las pequeñas y medianas empresas todo lo contrario.
Aquí es donde entra en juego el concepto de open-source routing.
Esta solución sería idónea ya que solo sería necesario un equipo común con una serie de características, que posteriormente se enunciarán, y un administrador que lo supervise.

Hay multitud de ofertas de open-source routing (Vyatta, XORP, Zebra, Quagga, …) que ofrecen servicios relacionados con el tema de enrutamiento (RIP, BGP, IGMP,…), servidores (HTTP, FTP, DNS,...) y seguridad (firewall, IDS, …).

## 1.2. Comparativa

A continuación se muestran algunas, de las muchas propuestas que existen en la actualidad. Muchas han quedado obsoletas u olvidadas debido a que no han logrado el éxito que se esperaba, por lo que han dejado de ser actualizadas.

**XORP**

XORP (eXtensible Open Router Platform) es una plataforma de código abierto que proporciona todas las funciones que implementan los protocolos de enrutamiento IPv4 e IPv6. Es la única plataforma que ofrece una capacidad integrada de multidifusión.

El desarrollo del proyecto fue fundado por Mark Handley en el año 2000. Recibió fondos de Intel y Microsoft y el software se lanzo al mercado por primera vez en julio del 2004. La última actualización se hizo en marzo del 2011, por lo que podemos comprobar que es un software que está en constante desarrollo. Esto es importante, ya que los errores que puedan surgir al usuario, se puedan solucionar en nuevas actualizaciones.

Entre sus principales objetivos de XORP, está la tarea de soportar todos los requisitos de los routers reales incluyendo los protocolos de enrutamiento, facilitar las APIs para crear extensibilidad en los protocolos de enrutamiento, mejorar el rendimiento del router y hacerlo robusto para que los paquetes no se colapsen o pierdan.

XORP puede convertirse en una alternativa de bajo coste en comparación a las costosas máquinas de Cisco Systems y otras empresas que dominan el mercado de enrutamiento, ya que se puede montar sobre PCs convencionales.

La base de código XORP consta de alrededor 670.000 líneas de código escritas en lenguaje C++ y desarrolladas principalmente en FreeBSD, pero totalmente compatibles con Linux y OpenBSD.

Según sus creadores soporta protocolos como RIP, BGP, OSPF en procesos independientes lo que significa que si un componente de software cae, no afectará a los otros que están utilizando el mismo hardware.

El proyecto soporta los siguientes protocolos de enrutamiento:

- Enrutamiento estático.
- RIP-2 y RIPng.
- BGP-4.
- OSPFv2 y v3.
- PIM-SM.
- IGMP v1, v2 y v3
- MLD v1 y v2.
- VRRP v2.

El lenguaje de línea de comandos es el modelo de la plataforma de Juniper Networks de JUNOS, el cual funciona bajo el sistema operativo Unix.
Por otra parte, hay algunas desventajas en XORP. Una de ellas es su rendimiento. Aunque el software supuestamente gestiona más de 700.000 paquetes por segundo en un PC estándar, la mayoría de los routers de alto rendimiento del mercado utilizan hardware especializado para acelerar el envío de paquetes.

XORP está disponible para su descarga como un Live CD o como código fuente a través de la página oficial del proyecto [1].

**ZEBRA** [2]

GNU Zebra es un software libre que gestiona protocolos de enrutamiento basados en TCP/IP. Es un proyecto libre, ya que forma parte del Proyecto GNU, y se distribuye bajo la Licencia Pública General GNU.

El proyecto Zebra se inició en 1996. La idea de Zebra vino originalmente de Kunihiro Ishiguro, quien había estado trabajando en NIS y en una empresa conjunta entre el ISP British Telecom y Marubeni. Cuando estaba trabajando para un ISP, se dio cuenta de una gran necesidad de un nuevo tipo de software de enrutamiento. Junto con Yoshinari Yoshikawa decidieron crear un nuevo motor de enrutamiento basado en licencia pública GNU.

Zebra es un software que permite montar routers sobre sistemas operativos tipo Unix. Este software dispone de una interfaz de configuración basada en el CISCO IOS, por lo que será útil a los administradores familiarizados con routers CISCO.

Tradicionalmente, la configuración de un router basado en UNIX se realizaba mediante los comandos ifconfig y route. El estado de las tablas se podía mostrar mediante la utilidad netstat. Estos comandos solamente se podían utilizar trabajando como root. Zebra, sin embargo tiene otro método de administración. En Zebra existen dos modos de usuario. Uno es modo normal y otro es el modo enable (habilitado). El usuario normal solo puede ver el estado del sistema y el enable puede cambiar la configuración del sistema. Como vemos esto se asemeja al modo de trabajo de los routers CISCO, en el que tendremos que entrar en el modo configuración para poder realizar los cambios.

El software tradicional de routing está compuesto por un programa o un proceso único que proporciona todas las funcionalidades de los protocolos de routing. Zebra sin embargo tiene una visión distinta. Está compuesto por una colección de varios demonios que trabajan juntos para construir una tabla. El demonio ripd maneja el protocolo RIP, mientras que el demonio ospfd controla el protocolo OSPFv2 y bgpd el protocolo BGP-4. Es sencillo añadir nuevos demonios de routing sin afectar a otros.

Los protocolos de enrutamiento que Zebra soporta son:

- RIPv1, v2, RIPng.
- BGP-4 y BGP-4 +.
- OSPFv2 y v3.

Esta opción no ofrece todos los servicios ni protocolos de enrutamiento necesarios para realizar las prácticas de LPR, y por lo tanto no se podría afirmar que este software podría sustituir completamente a los router CISCO. Desde su última actualización en Septiembre del 2005 el desarrollo de Zebra se ha detenido para dar lugar a su nuevo sucesor, Quagga.

**QUAGGA** [3]

El proyecto lleva ese nombre debido a una subespecie extinta de cebra africana. Como se ha comentado anteriormente, Quagga es el sucesor de Zebra.

Al igual que en Zebra, la arquitectura multiproceso permite un sistema más fácilmente extensible, gestionado y modular. Quagga copia el estilo del modo de trabajo del CISCO IOS y tiene un proceso independiente para cada protocolo. Un sistema con Quagga instalado actúa como un router dedicado. Quagga intercambia información con otros routers mediante protocolos de routing, utilizándola para actualizar el núcleo de las tablas de routing de forma que la información correcta esté en el lugar correcto. Quagga permite la configuración dinámica y es posible ver la información de la tabla de routing desde el interfaz de terminal de Quagga.

El proyecto soporta los siguientes protocolos de enrutamiento:

- RIPv1, v2, RIPng.
- BGP-4 y BGP-4 +.
- OSPFv2 y v3.

Además de soportar ipv4, también soporta ipv6. Posee una arquitectura avanzada que le proporciona una gran calidad y potencia, con un motor multiservidor de encaminamiento.
Quagga tiene un interfaz de usuario interactivo para cada protocolo de routing.

Está diseñado especialmente para NetBSD, FreeBSD, Solaris y Linux y solamente se tendrá que instalar el paquete quagga.

A diferencia de otros open-source routing, sólo se puede descargar los paquetes, que posteriormente se tendrán que instalar en una distribución de Linux, los cuales actuarán como demonios independientes, en lugar de una distribución Linux como ocurría en XORP. Aunque se podría crear un LiveCD de un Linux con Quagga, esto podría ser un inconveniente para alguien que no tenga mucha experiencia.

Este software es una opción interesante debido a su estructura interna e instalación por paquetes, aunque ofrece los mismos servicios que su antecesor Zebra.

**BIRD** [4]

El nombre de “BIRD” es un acrónimo de ‘Internet Routing Daemon’ que significa demonio de encaminamiento de internet.

BIRD se desarrolló en la facultad de matemáticas y física de la Universidad Charles en Praga, como un proyecto estudiantil en 1999. El proyecto estuvo parado un tiempo hasta el 2003. Puede ser libremente distribuido bajo los términos de la Licencia Publica General GNU y es una alternativa a Quagga/Zebra.

BIRD ha sido diseñado para trabajar en todos los sistemas tipo UNIX, FreeBSD, NetBSD y OpenBSD.

BIRD es un proyecto cuyo cometido es el desarrollo de un completo demonio de ruteo de IP, cuyo modo de acceso es a través de línea de comando y diseñado especialmente para ajustarse a cualquier sistema operativo de tipo UNIX. BIRD es compatible con el protocolo IPv4 o IPv6, aunque tendrán que ser compilados de forma separada ya que no son compatibles al mismo tiempo.

Protocolos de enrutamiento soportados:

- Rutas estáticas.
- RIPv2.
- BGPv4.
- OSPFv2 y v3.

BIRD brinda un mecanismo de configuración realmente flexible y fácil de maniobrar, basado en un fichero de configuración, que se activará mediante un SIGHUP.

En definitiva, BIRD ofrece un potente lenguaje para la aplicación de filtros de "ruteo" de IP dinámica, pero también de rutas estáticas, y con múltiples tablas de routing, ideal para dominar el conjunto de protocolos de "ruteo" vigentes en la red de redes hoy en día.

Al igual que los proyectos anteriores, BIRD no nos ofrece todos los servicios que nos podría ofrecer un router CISCO, como por ejemplo la seguridad. Además, al igual que Quagga, solo están disponibles los paquetes, y las actualizaciones, que tendremos que compilar e instalar en nuestra distribución Linux.

## 1.3. Vyatta

Vyatta [5] ha creado una de las mejores soluciones de red, la cual está optimizada para ofrecer seguridad integrada y funcionalidad de enrutamiento para entornos físicos y virtuales.

El sistema Vyatta se diseñó con el propósito de reemplazar las versiones comerciales de Cisco, desde la serie 1800 hasta la 7200, apoyándose en el bajo coste y flexibilidad de las soluciones open source, y que además al estar basado en Linux funcionaría en la mayoría de sistemas basados en la tecnología x86.

En la versión VC3, Vyatta empieza a utilizar XORP, en concreto su modelo de pila de enrutamiento, y es patrocinado por éste durante 2 años. Más adelante, en la versión VC4, el equipo de Vyatta toma la decisión de ir en otra dirección. A partir de ahí ya no

deciden utilizar XORP, porque quieren dotar a Vyatta de más funciones (firewall, VPN, DHCP, VLAN, etc) las cuales XORP no dispone.
A principios de 2007, Vyatta tenía un gran número de clientes, los cuales comienzan a utilizar el sistema en redes de proveedores de servicios, en los que había que implementar el protocolo BGP. Como el despliegue de redes era tan grande, se encontraron con una serie de limitaciones en el código fuente de XORP relacionados con la escalabilidad y el rendimiento. Para solucionar esto, los ingenieros de Vyatta hicieron muchas reformas en el código fuente de XORP, y al final se determinó que los costes eran mayores que si lo desarrollaban ellos mismos, por lo que se inició el trabajo para realizar un nuevo diseño de pila de enrutamiento. Así nació la versión VC4 con la colaboración de Quagga. Después de varias pruebas, consiguieron la escalabilidad, el rendimiento y la estabilidad que estaban buscando. Ahora bien, hay algunas funciones que tenía XORP que la nueva pila no tiene, como por ejemplo algunas características de multidifusión. Por otro lado, mejoraron el manejo de la conexión de rutas estáticas cuando un enlace se cae.

Vyatta ofrece dos opciones de uso, los dispositivos hardware o el software Vyatta Network OS.
Los aparatos hardware, integran el software Vyatta Network OS y software de seguridad. Estos dispositivos están basados en las plataformas estándar de hardware x86, y eliminan cualquier dependencia de hardware propietario. En concreto, ofrecen 4 tipos de dispositivos hardware de diferentes características técnicas, que varían según las necesidades del comprador.

Por otro lado, también ofrecen el software Vyatta Network OS en 3 modelos, Vyatta Subscription Edition (VSE), Vyatta Core (VC) y Vyatta Plus.
En este proyecto nos centraremos en la versión VC, ya que es la única opción de código abierto y por tanto gratuita. Esta versión no se recomienda para la redes de entornos de producción y no dispone de algunos servicios profesionales, como por ejemplo configurar los interfaces serie, DSL y wireless, configurar los encapsulamiento (Cisco HDLC, Frame Relay, Classical IPoA, Bridged Ethernet) de éstos, configurar algunas características de VPN (OpenVPN), ni servicios como el acceso remoto, ya que solo están disponibles en la versión de pago (Vyatta Subscription).
VSE, es una versión comercial, la cual se puede pedir una versión de prueba de 30 días, solamente si luego les aseguras que estarás dispuesto a comprarla. El modelo comercial, además de los servicios de la versión VC, ofrece un servicio de actualizaciones, soporte técnico y formación.
Y con Vyatta Plus, además de la todas las características de VSE, incluye los servicios de Vyattaguard (filtrado de URL de base de datos) y Snort VRT service (servicio comercial de IPS).

Vyatta se puede instalar tanto sobre un servidor físico como en una máquina virtual, permitiendo disponer de servicios de seguridad de red en entornos virtualizados y entornos en nube. Vyatta ofrece a los proveedores de los servicios de nube la opción de asegurar y gestionar sus complejas redes ofreciendo mucho más que un simple firewall, en concreto, VPN con IPsec, SSL en OpenVPN, prevención de intrusiones de red, seguridad de filtrado web y mucho más.

El sistema Vyatta ofrece las siguientes funcionalidades:

Servicios IP [6]:

- SSH.
- Telnet.
- DHCP.
- DNS.
- NAT [7].
- Web caching.

Enrutamiento básico:

- Políticas de enrutamiento [8].
- RIP [9].
- BGP [10].
- OSPF [11].

Servicios de seguridad:

- Firewall: IPv4, IPv6 y por zonas [12].
- Detección de intrusiones (snort) y filtrado web [13].

Servicios de Red Privada Virtual (VPN) [14]:

- VPN Lan-to-Lan con IPSec.
- Acceso remoto con PPTP.
- Acceso remoto con L2TP e IPSec.
- VPN Lan-to-Lan y acceso remoto con OpenVPN.

QoS (Quality of Services) [15]:

- Traffic shape: permite controlar y restringir la anchura de banda (bandwith) de las conexiones salientes.
- Traffic límite: permite controlar y restringir la anchura de banda (bandwith) de las conexiones entrantes.

Servicios de alta disponibilidad:

- WAN Load balancing: permite encriptar el tráfico saliente a través de diferentes proveedores.
- VRRP (Virtual Router Redundancy Protocol): permite que un cluster de Vyattas actúen como un único dispositivo.
- Clustering: permite disponer de alta disponibilidad entre dos servidores Vyatta.
- Statefull NAT and Firewall Failover: permite replicar el estado de las conexiones entre los servidores de un cluster, de manera que en caso de error de un nodo, las conexiones de mantienen activas.
- RAID 1: permite implementar RAID 1 de discos, tanto para hardware como para software.
- Configuration Synchronization: sincroniza la configuración entre dos nodos en alta disponibilidad.

Las características mínimas para instalar el software Vyatta en un PC son:

- Procesador: Pentium III a 500 MHz
- Memoria RAM: 384 MB
- Disco Duro: 10 Gb

Como podemos comprobar las características mínimas para la instalación del software de Vyatta son más exigentes que las demás opciones.

Al igual que las demás opciones open-source, todas las funciones son fácilmente configurables mediante el uso de intuitivas instrucción desde la "línea de comandos", y además incluye una interfaz gráfica web (GUI).

## 1.4. Tabla Comparativa

A continuación se muestra una tabla en la que se ven reflejadas las distintas características de los routers open source y una versión del dispositivo comercial más popular del mercado, CISCO.

| | XORP 1.8.3 | QUAGGA 0.99.18 | BIRD 1.3.2 | VYATTA VC 6.2 | CISCO 2651 |
|---|---|---|---|---|---|
| | | | | | |
| Sistema Operativo | Linux | Unix-like | Unix-like | Linux | Cisco IOS |
| Opensource | SI | SI | SI | SI | NO |
| Instalación | SI/NO | SI | SI | SI/NO | NO |
| | | | | | |
| Rutas Estáticas | SI | SI | SI | SI | SI |
| RIPv2 | SI | SI | SI | SI | SI |
| OSPFv3 | SI | SI | SI | SI | SI |
| BGPv4 | SI | SI | SI | SI | SI |
| | | | | | |
| NAT | SI | SI | NO | SI | SI |
| VRRP | SI | NO | NO | SI | SI |
| Mapas de rutas | SI | SI | SI | SI | SI |
| VPN IPSec | NO | NO/Linux | NO | SI | SI |
| VPN SSL | NO | NO/Linux | NO | SI | NO |
| | | | | | |
| Cliente FTP | NO/Linux | NO/Linux | NO/Linux | SI | SI |
| Cliente TFTP | NO | NO | NO | SI | SI |
| Telnet | SI | SI | SI | SI | SI |
| Servidor SSH | NO/Linux | NO/Linux | NO/Linux | SI | SI/NO |
| Servidor HTTP | NO | NO | NO | SI | SI |
| | | | | | |
| Servidor DHCP | NO/Linux | NO/Linux | NO/Linux | SI | SI |
| Servidor NTP | NO/Linux | NO/Linux | NO/Linux | NO | SI/NO |
| Cliente NTP | NO/Linux | NO/Linux | NO/Linux | SI | SI |
| SNMP | NO | SI | NO | SI | SI |
| | | | | | |
| Ping | SI | SI | SI | SI | SI |

| | | | | | |
|---|---|---|---|---|---|
| Traceroute | SI | SI | SI | SI | SI |

NOTAS:
Unix-like: es un sistema operativo que se comporta de manera similar a un sistema Unix, aunque no necesariamente conforme a su certificación.
No/Linux: significa que el software en cuestión no trae esa opción, pero al funcionar sobre el sistema operativo Linux, éste sí que dispone de esa característica.

Como conclusión hay decir que Vyatta posee las mismas funcionalidades que XORP, Quagga y Bird, e incorpora en el software servicios IP como ssh, telnet, dhcp, dns, nat, ftp, tftp. Al ser Quagga y Bird procesos de enrutamiento, no es de esperar que incluyan estos servicios, ya que se supone que los podremos instalar en la máquina que actuará como router.

Además proporciona servicios de red privada virtual (VPN), QoS, VRRP, seguridad firewall y detección de intrusiones. Todo esto hace que Vyatta sea un duro competidor para CISCO, ya que posee casi todas las funcionalidades de éste.

Al final, después de buscar y comparar las diferentes alternativas de software open-source routing, Vyatta es una, o quizás la mejor opción para disponer de un router de manera gratuita (solo nos hace falta un ordenador con unas características mínimas) y con una amplia gama de posibilidades para el ruteo de datos y la protección en la red.
Con Vyatta se puede desplegar arquitecturas complejas de red a un coste muy competitivo comparado con CISCO o Juniper por ejemplo. Además, si se necesita de un apoyo, siempre se puede contratar una suscripción de soporte.

Por otro lado, ya son miles los usuarios que han decidido utilizar Vyatta, y cuenta con una comunidad de cientos de usuarios certificados y con muchos renombrados negocios entre sus clientes comerciales. Todo esto hace que cualquier problema que pueda surgir, se pueda compartir con otras personas para su rápida solución.

Debido a todo esto, me dispongo a probar las diferentes opciones del software Vyatta para su posterior comparación con los dispositivos CISCO.

# 2.- VYATTA

## 2.1. Modos de Uso

A continuación se muestran los principales modos de utilización con los que se podrá arrancar e instalar el sistema Vyatta:

- **LiveCD**: se puede arrancar el sistema Vyatta directamente desde un LiveCD sin necesidad de realizar ninguna instalación en nuestra maquina. Este método es usado para realizar evaluaciones y pruebas sin preocuparnos de la instalación.

- **Instalación física**: Vyatta puede ser instalado y funcionar en la mayoría de PCs y servidores de arquitectura x86. Puede ser instalado en discos duros, y memorias flash.

- **Instalación en maquina virtual**: Como última alternativa de instalación, Vyatta puede funcionar en una maquina virtual, permitiendo virtualizar la red. En concreto el software Vyatta ha sido optimizada para VMware y XenServer. Estas plataformas proveen un alto grado de flexibilidad en las configuraciones del sistema, especialmente en la memoria y en las interfaces Ethernet.

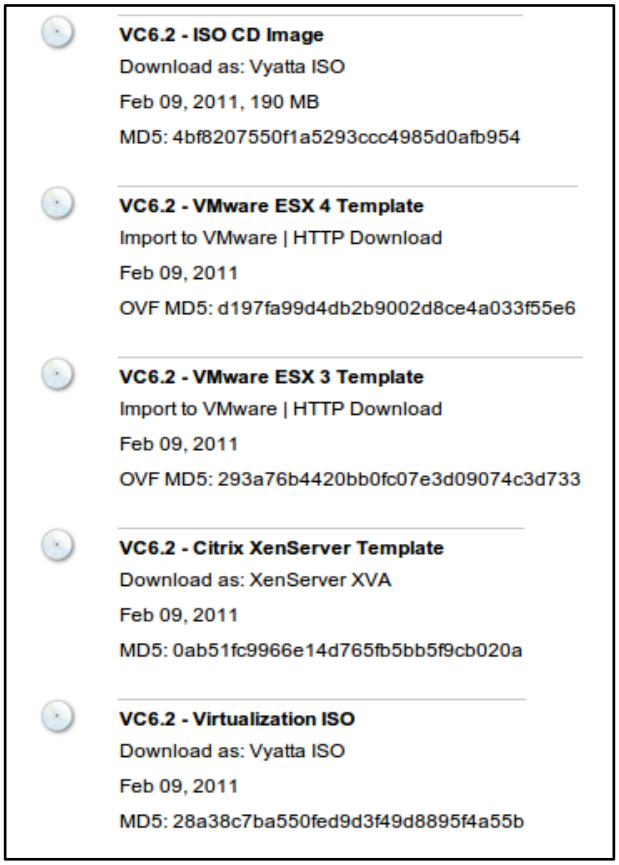


**Figura 1 – Modos descargas**

Desde la página oficial de Vyatta, se puede descargar de manera gratuita la opción que mejor se acerque a las necesidades de cada uno.

## 2.2. Interfaz Vyatta

Se puede interactuar con Vyatta de dos maneras diferentes:

1. Mediante línea de comandos (CLI)
2. Mediante una interfaz grafica basada en web (GUI)

### 2.2.1. Interfaz de línea de comandos (CLI)

Incluye dos tipos de comandos, por un lado tenemos los que son específicos de Vyatta, los cuales nos permiten configurar y operar con el sistema Vyatta y por otro lado, comandos propios de la Shell del Linux. Todos los usuarios tienen acceso a los comandos, aunque dependiendo de los privilegios de su cuenta podrán o no realizar ciertas acciones.

Dentro de los comandos de Vyatta, hay dos modos, el operacional y el de configuración.

El **modo operacional**, provee acceso para mostrar información, activar o desactivar la depuración, realizar la configuración del terminal, cargar y guardar la configuración y reiniciar el sistema.

El **modo de configuración**, permite crear, modificar, eliminar y mostrar la información de configuración, así como comandos para navegar por la jerarquía de configuración.

Para entrar en el modo de configuración, se usará el comando *configure* como se ve en la figura 2. Escribiendo *exit*, se volverá al modo operacional.

```
vyatta@vyatta:~$ configure
[edit]
vyatta@vyatta#
```

**Figura 2 – Modo configuración**

Se puede obtener la lista de los posibles comandos operacionales pulsando una vez el tabulador. Si lo pulsamos dos veces seguidas, como se puede observar en la figura 3, se obtendrá una breve descripción de los comandos al lado de éstos.

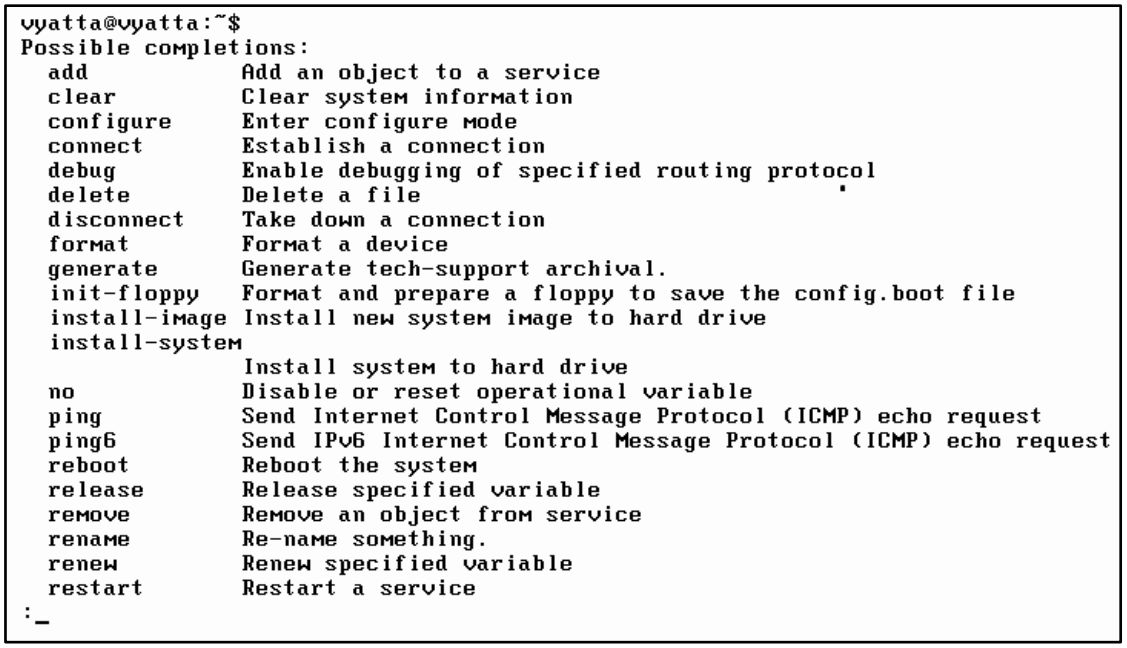


**Figura 3 – Comandos Operacionales**

Por otra parte, en la figura 4, se pueden ver los comandos de configuración. A diferencia de los comandos operacionales, éstos son propios del software Vyatta. Al igual que en modo operacional se puede ver una descripción de los comandos disponibles.

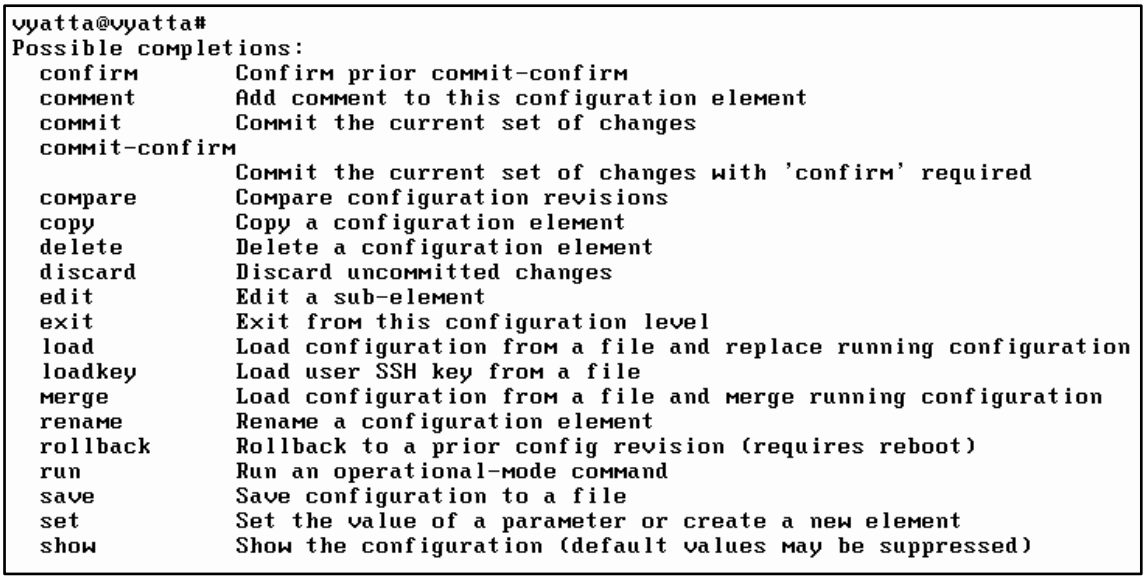


**Figura 4 – Comandos Configuración**

### 2.2.2. Interfaz grafica basada en web (GUI)

La interfaz gráfica es la otra alternativa para interactuar con el sistema Vyatta. La interfaz gráfica está desactivada por defecto por razones de seguridad. Si se quiere usar, primero se tendrá que habilitar el servicio https en la interfaz de línea de comandos.

```
vyatta@vyatta# set service https
[edit]
vyatta@vyatta# commit
[ service https ]
Generating a 1024 bit RSA private key
.......++++++
.........................................++++++
writing new private key to '/etc/lighttpd/server.pem'
-----

[ service https ]
Stopping web server: lighttpd.
Starting web server: lighttpd.
Stopping PAGER server
Starting PAGER server
```

**Figura 5 – Servicio https**

Para acceder a la interfaz gráfica, habrá que ir a un navegador web e introducir la dirección IP que hayamos definido a nuestro interfaz Vyatta.
En la figura 6 se puede ver que el modo de acceso es tan sencillo como introducir la dirección IP de la interfaz del router en la barra de direcciones del navegador web, e introducir nuestro usuario y contraseña. Como credenciales se utilizarán las mismas que en la interfaz de línea de comandos.

La interfaz gráfica hace que todo sea más gráfico e intuitivo para una persona que no está acostumbrada a la interfaz de línea de comandos, ya que en estos tiempos lo que prima es la sencillez de las cosas y lo visualmente agradable.

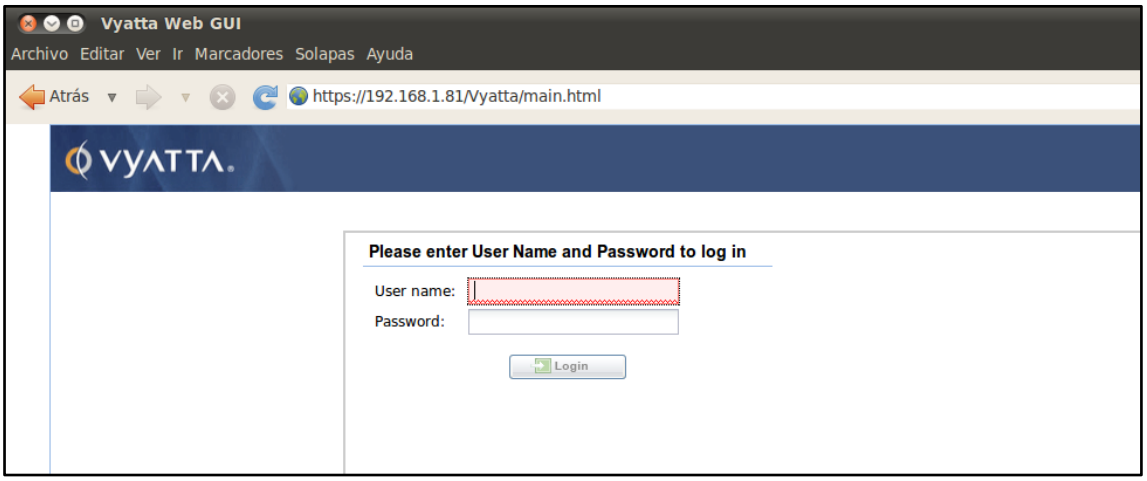


**Figura 6 – Interfaz acceso**

Una vez dentro las opciones de configuración son las mismas que las de la línea de comandos. Todo que se podía realizar mediante la interfaz de línea de comandos, se podrá realizar mediante la interfaz gráfica.

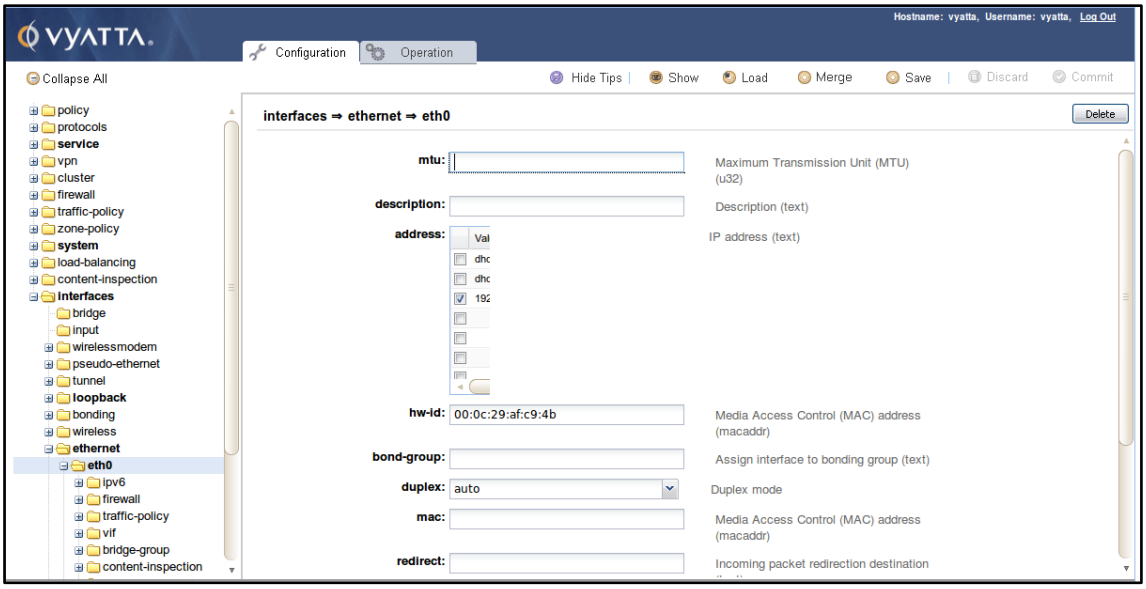


**Figura 7 – Opciones interfaz gráfica**

## 2.3. Creación y Administración de usuarios

Por defecto, el sistema tiene una cuenta de usuario predefinida, el usuario ***vyatta***. La contraseña que tiene asignada esta cuenta es también ***vyatta*.** Este usuario, tiene privilegios de administrador y puede ejecutar todos los comandos de Vyatta y del sistema operativo.

El sistema Vyatta soporta dos tipos de usuario, el Administrador y el Operador.

### 2.3.1. Administrador

Los usuarios administradores, tienen acceso total a la interfaz de línea de comandos. Pueden ver, configurar, borrar información y ejecutar todos los comandos operaciones de Vyatta. También pueden ejecutar todos los comandos Shell del sistema operativo.

Como se ha comentado anteriormente, el usuario por defecto vyatta, es un usuario administrador. Para crear un usuario administrador, habrá que seguir los siguientes pasos en el modo configuración:

> ***vyatta@vyatta# set system login user*** *nombre-usuario* ***level admin***
> ***vyatta@vyatta# set system login user hector authentication plaintext-password*** *contraseña*
> ***vyatta@vyatta# commit***

donde *nombre-usuario* es el identificador de la nueva cuenta de usuario que se quiera crear y *contraseña* la que se quiera asignar.

Primero se creará un nuevo usuario dándole el nivel de administrador, y después una contraseña en texto plano, aunque en los archivos de configuración aparecerá encriptado.

```
vyatta@vyatta# show system login user administrador
  authentication {
      encrypted-password $1$BdcveYTf$CB1EiUSSjpW1DSc0N3KM10
      plaintext-password ""
   }
   level admin
```

### 2.3.2. Operador

Los operadores tienen acceso de solo lectura, lo que significa que no podrán realizar ninguna modificación en los archivos de configuración. Por otro lado si pueden ejecutar comandos del sistema operativo, por lo que podrán configurar las opciones del terminal y salir de la línea de interfaz de comandos. Aunque no puedan entrar en el modo de configuración, sí que la podrán ver cómo están hechos los cambios mediante la instrucción ***show configuration*** en el modo operacional.

Para crear un usuario Operador se procederá de la siguiente manera:

```
vyatta@vyatta# set system login user nombre-usuario level operator
vyatta@vyatta# set system login user nombre-usuario authentication
plaintext-password contraseña
vyatta@vyatta# commit
```

Los pasos son los mismos que para crear un usuario administrador, con la diferencia de que ahora el nivel será *operator*.

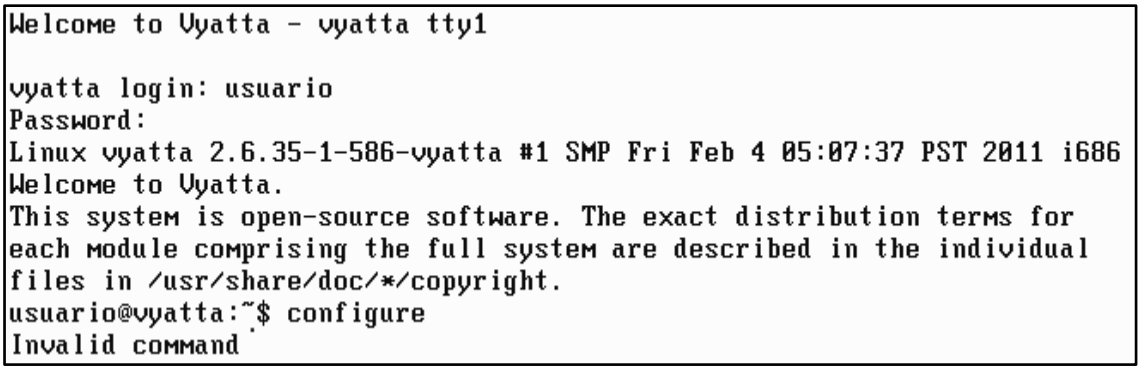


**Figura 8 – Pantalla bienvenida**

En la figura 8 se ve que si se intenta acceder con la cuenta de usuario (Operador) al comando configure, el sistema Vyatta lo impide (*Invalid command*), ya que este usuario no tiene los privilegios suficientes para acceder al modo de configuración.

## 2.4. Guardar Cambios

En el sistema Vyatta, los cambios realizados en modo configuración no son efectivos, hasta que se ejecuta la instrucción ***commit***. Al ejecutar esta instrucción, todos y cada uno de los cambios que se hayan realizado se guardarán automáticamente.

Los cambios que no se han confirmado todavía aparecen con el símbolo '+' por delante si hemos añadido algo (set), con el símbolo '-', si hemos borrado (delete) o con el símbolo '>', si se ha modificado algo.

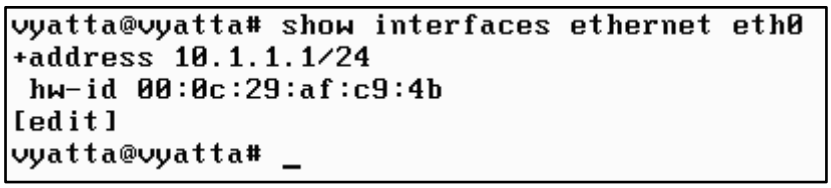


**Figura 9 – Interfaces**

Como se puede ver en este ejemplo, se ha asignado una dirección IP al interfaz eth0, pero no se ha confirmado con commit, por lo que delante del cambio que se ha hecho aparece el símbolo '+'.

## 2.5. Descartar Cambios

Una vez realizado los cambios, si se desea descartarlos, en vez de ir borrando uno a uno, se pueden deshacer todos los que se hayan hecho en la sesión.
Para ello se utiliza la instrucción ***discard***.

```
vyatta@vyatta# show interfaces ethernet eth0
+address 10.1.1.1/24
+description "Interfaz eth0"
 hw-id 00:0c:29:af:c9:4b
[edit]
vyatta@vyatta# discard
Changes have been discarded
[edit]
vyatta@vyatta# show interfaces ethernet eth0
 hw-id 00:0c:29:af:c9:4b
[edit]
vyatta@vyatta# _
```

**Figura 10 – Descartar cambios**

En la figura 10 se puede comprobar lo comentado anteriormente. Se han realizado unos cambios de adición en la configuración del interfaz Ethernet eth0, pero que aún no han sido guardados, ya que aparecen con el símbolo '+'. Posteriormente se ha ejecutado la instrucción discard y los cambios que estaban pendientes de guardar se han descartado.

Hay que decir que si se ha realizado algún cambio en el modo de configuración, no se podrá salir al modo operacional sin antes guardar o deshacer los cambios.

## 2.6. Salvar Configuración

Toda la configuración que se realice en el sistema Vyatta puede ser guardada usando el comando ***save***. Por defecto la configuración es guardada en el archivo ***config.boot***.

Si se ha instalado el sistema, el directorio de configuración será ***/opt/vyatta/etc/config***. En cambio si se está ejecutando el sistema mediante un LiveCD, el directorio será ***/media/floppy/config***.

```
vyatta@vyatta# save
Saving configuration to '/opt/vyatta/etc/config/config.boot'...
Done
[edit]
vyatta@vyatta#
```

**Figura 11 – Salvar cambios**

No hay que confundir este comando con la instrucción commit. La instrucción save, guarda solo los cambios confirmados con commit. Si se tratan de guardar los cambios que no han sido confirmados con commit, el sistema nos avisará de que solo serán guardados los cambios que hayan sido confirmados.

Por otra parte también hay que tener en cuenta, que si se reinicia el sistema solo se guardarán aquellos cambios que se hayan guardado con la instrucción save, y por lo tanto estén en el archivo config.boot, ya que será éste el que se cargue al iniciar el sistema.

También se puede guardar la configuración en un archivo diferente, pero tendrá que ser en el directorio por defecto. Esto se puede hacer especificando en la instrucción *save* un nombre de archivo.

```
vyatta@vyatta# save nuevo_archivo
```

Mediante esta instrucción se guardará toda la configuración en /opt/vyatta/etc/config/nuevo_archivo, por lo que si se quiere guardar la configuración en un directorio diferente, tan sólo habrá que escribir *save* y la ruta donde guardarlo.

Esta técnica hace que sea posible tener múltiples archivos de configuración para las diferentes situaciones, como por ejemplo si se están haciendo pruebas para más adelante implementarlas definitivamente.

## 2.7. Cargar Configuración

Para cargar un archivo de configuración que ya se tenía previamente guardado, habrá que utilizar el comando ***load***.

```
vyatta@vyatta# load
Loading configuration from '/opt/vyatta/etc/config/config.boot'...
No configuration changes to commit
[edit]
vyatta@vyatta# _
```

**Figura 12 – Cargar configuración**

Primero se cargará el archivo y luego se salvarán los cambios,

```
vyatta@vyatta# load
vyatta@vyatta# save
```

o si lo el archivo era uno que se había creado anteriormente,

```
vyatta@vyatta# load archivo_nuevo
vyatta@vyatta# save
```

Después esto, el siguiente paso es ver cómo configurar las características del sistema, entre las que se encuentran, el nombre de host, de dominio, ruta por defecto y alias.

## 2.8. Configuración Inicial

En este apartado, se realizarán las configuraciones básicas iniciales del sistema Vyatta. Para ello, se utilizara el escenario de la figura 13.

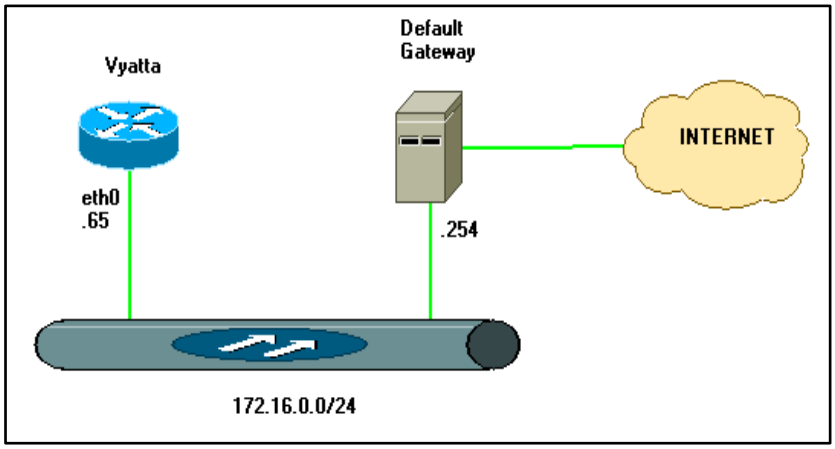


**Figura 13 – Configuración básica**

### 2.8.1. Nombre del Host

El nombre del sistema Vyatta es establecido mediante la instrucción ***system host-name*** y puede incluir letras, números y guiones.
Para establecer el nombre de R1 al host del escenario de la figura 13 se ejecutará:

```
vyatta@vyatta# set system host-name R1
vyatta@vyatta# commit
```

Para comprobar que realmente el nombre ha sido cambiado:

```
vyatta@R1# show system host-name
  host-name R1
```

### 2.8.2. Dominio

El nombre del domino del sistema se establece mediante la instrucción ***system domain-name.*** Al igual que el nombre del host, puede incluir letras, números y guiones.

Se establecerá el nombre de dominio midominio.com así:

```
vyatta@R1# set system domain-name midominio.com
vyatta@R1# commit
```

Seguidamente se comprueba el cambio:

```
vyatta@R1# show system domain-name
  domain-name midominio.com
```

### 2.8.3. Ruta por defecto

Para especificar la ruta por defecto que tiene que tomar el router en caso de que no sepa que camino escoger, se procederá de la siguiente manera:

```
vyatta@R1# set system gateway-address 172.16.0.254
vyatta@R1# commit
```

Para ver los cambios realizados:

```
vyatta@R1# show system gateway-address
  gateway-address 172.16.0.254
```

### 2.8.4. Alias

Se puede definir uno o varios alias, para mapear la dirección IP del sistema a más de un nombre de host. En este escenario se procedería de la siguiente manera:

```
vyatta@R1# set system static-host-mapping host-name R1 alias router1
vyatta@R1# commit
```

Para ver los cambios:

```
vyatta@R1# show system static-host-mapping
  host-name  R1  {
       alias  router1
  }
```

## 2.9. Comandos Básicos

En esta sección se describirán los comandos básicos, los cuales se emplearán para la realización de las prácticas. De esta forma, se conseguirá una familiarización con los comandos básicos para que luego resulten familiares y en caso de duda se pueda resolver rápidamente.

### 2.9.1. Interfaces

A continuación se detallarán los comandos con los cuales configuraremos las opciones de los interfaces.
Para ello, en modo de configuración, se utilizará la instrucción ***set interfaces*** seguida de los siguientes parámetros:

- ***set interfaces serial:*** para configurar los parámetros de los interfaces serie. En la versión que se utiliza para este proyecto no se permite su configuración.
- ***set interfaces wireless:*** esta instrucción permite la configuración de los interfaces wireless. Los dos modos de operación que existen para este interfaz son como punto de acceso wireless (WAP) y como estación. En el modo WAP, se pueden configurar opciones como el tipo de seguridad, la contraseña de la red, el canal, el identificador, etc. Y mediante el modo estación, el sistema actúa como un cliente el cual accede a la red a través de un WAP.
- ***set interfaces ethernet*** *ethX:* esta es la instrucción que más se utilizará por realizar la totalidad de las prácticas con estos tipos de interfaces. Se podrá configurar diferentes parámetros como:
    - ***address [****IP/netmask****]:*** establece una dirección IP y una máscara de red al interfaz.
    - ***description [****descripción****]:*** especifica una breve descripción del interfaz.
    - ***disable:*** deshabilita el interfaz ethernet sin borrar la configuración.
    - ***disable-link-detect:*** deshabilita la detección física del interfaz, ya que el sistema detecta cuando un interfaz está conectado o desconectado, habilitando su interfaz.
    - ***duplex [****duplexity****]:*** establece el modo dúplex, en donde duplexity puede ser *auto* (opción por defecto, en donde el router negocia

automáticamente el modo con el otro interfaz), *half* (half dúplex) o full (full dúplex).

- ***firewall [****reglas****]:*** aplica las reglas firewall al interfaz.
- ***hw-id [****dirección-mac****]:*** asocia el nombre del interfaz con una dirección MAC. Cuando el sistema arranca, si no se ha especificado el parámetro hw-id, este le establece una. Si un hw-id es especificado, el interfaz es asociado con esa NIC. Esta instrucción es usada cuando una nueva NIC es añadida al sistema o si se quiere asignar un nombre especifico al interfaz.
- ***ip enable-proxy-arp:*** habilita el proxy ARP en el interfaz.
- ***ip rip:*** habilita RIP en un interfaz.
  - ***authentication*** *contraseña:* especifica la autenticación RIP del interfaz. Contraseña puede ser una clave md5 o en texto plano.
  - ***split-shorizon [****disable* | *poison-reverse****]:*** habilita el mecanismo de split-horizon poison-reverse o lo deshabilita.
- ***mac [****dirección-mac****]:*** establece una dirección MAC al interfaz. Por defecto el interfaz viene con una dirección MAC de fábrica. Si se usa este comando se sobrescribirá la opción hw-id el cual es la dirección MAC de fábrica de la NIC. No en todas las tarjetas se podrá cambiar la dirección.
- ***mtu [****valor-mtu****]:*** especifica el valor MTU del interfaz.
- ***speed [****velocidad****]:*** establece la velocidad del interfaz. Por defecto el interfaz auto negocia la velocidad. El valor de la velocidad, en Mbps, puede ser auto, 10, 100 ó 1000.

### 2.9.2. Services

Otras instrucciones que se usarán mucho son las que se utilizan para configurar los servicios. Para ellos se ejecutará ***set service*** seguido de las siguientes opciones:

- ***dhcp-relay interface [****interfaz****]:*** especifica el interfaz a usar para aceptar peticiones DHCP o retransmitir mensajes de clientes DHCP.
- ***dhcp-server:*** habilita al sistema para que funcione como un servidor DHCP.
- ***dhcpv6-relay listen-interface [****interfaz****]:*** especifica el interfaz a usar para aceptar peticiones DHCPv6.
- ***dhcpv6-server:*** habilita la funcionalidad de servidor DHCPv6.
- ***dns dynamic interface [****intefaz****]:*** habilita el servicio DDNS en el interfaz.
- ***dns forwarding system:*** especifica el reenvío a los servidores DNS del sistema de nombres configurado.
- ***https:*** habilita el acceso mediante la interfaz gráfica GUI.
- ***lldp:*** habilita el servicio LLDP.
- ***nat:*** habilita el servicio NAT.
  - ***rule [****num-regla****]:*** define una regla NAT.

        - ***disable:*** deshabilita la regla num-regla.
        - ***inbound-interface [****interfaz****]:*** especifica la interfaz por donde se recibirá el tráfico entrante de una regla DNAT.
        - ***inside-address [****dirección****]:*** define la dirección interior para una regla DNAT.
        - ***outbound-interface [****interfaz****]:*** especifica el interfaz por donde será transmitido el tráfico saliente de origen.
        - ***outside-address [****dirección****]:*** define la dirección exterior para una regla SNAT.
        - ***type [****tipo****]:*** establece el tipo de traducción para una regla NAT. Éste puede ser source, destination ó masquerade.
- ***ssh:*** habilita el servicio SSH como un protocolo de acceso en el sistema Vyatta.
    - ***allow-root:*** especifica que el acceso mediante el usuario root está permitido en las conexiones SSH.
    - ***disable-password-authentication:*** permite a los usuarios que accedan al sistema por ssh no tener que autenticarse.
    - ***listen-address [****dirección****]:*** permite el acceso SSH a esa dirección.
    - ***port [****puerto****]:*** especifica el puerto que el sistema usará para el servicio.
    - ***protocol-version [****versión****]:*** especifica qué versión de SSH está habilitada, pudiendo ser v1, v2 o ambas. Por defecto la v2 es la habilitada.
- ***telnet:*** habilita el servicio telnet como protocolo de acceso al sistema.
    - ***allow-root:*** especifica que el acceso mediante el usuario root está permitido en las conexiones telnet.
    - ***listen-address [****dirección****]:*** configura el acceso a telnet en una dirección específica.
    - ***port [****puerto****]:*** especifica el puerto que el sistema usará para el servicio.
- ***snmp:*** define la comunidad SNMP y obtiene información para el sistema.
- ***webproxy listen-address [****dirección****]:*** especifica una dirección de escucha webproxy.

### 2.9.3. Protocols

Con las instrucciones que se mostrarán a continuación, se podrá configurar protocolos como bgp, ospf, rip, rutas estáticas, etc. Para ello se usará la instrucción ***set protocols*** seguido de los siguientes parámetros:

- ***bgp [****asn****]:*** crea una instancia BGP en el router, y lo localiza dentro de un sistema autónomo (AS).
- ***ospf:*** habilita el protocolo OSPF en el sistema Vyatta.
- ***rip:*** habilita el protocolo RIP para la configuración de las opciones.
    - ***network [****direccion-red****]:*** especifica una red para el protocolo RIP.

- ***redistribute connected:*** permite redistribuir las rutas conectadas directamente a las tablas de enrutamiento RIP.
- ***interface [****interfaz****]:*** habilita RIP en el interfaz especificado.
- ***timers timeout [****segundos****]:*** permite establecer el tiempo que tarda el sistema en actualizar su tabla de rutas.
- ***timers update [****segundos****]:*** permite establecer cuanto tiempo tarda el router en volver a anunciar su tabla de rutas.

- ***static route [****subnet****] next-hop [****dirección****]:*** permite configurar el siguiente salto de una ruta estática estableciendo la subnet a la que se quiere acceder y por cual dirección deberá ir el paquete.

### 2.9.4. Extras

Existen también unos comandos que no están en ningún manual, pero que ofrecen algunas funciones que cabe destacar.

Como se dijo en el capítulo anterior, desde la versión VC4, el sistema Vyatta se basa en Quagga, el cual es el responsable de las operaciones de los protocolos de enrutamiento y del mantenimiento del RIB. Mediante la instrucción ***aptitude show vyatta-quagga***, se podrá ver la versión de Quagga sobre la que se basa Vyatta.

```
Package: vyatta-quagga
State: installed
Automatically installed: no
Version: 0.99.17-15+mendocino1
Priority: extra
Section: contrib/net
Maintainer: Vyatta Package Maintainers <maintainers@vyatta.com>
Uncompressed Size: 5370 k
Depends: libc6 (>= 2.11), libcap2, libncurses5 (>= 5.7+20100313), libpam0g (>=
         0.99.7.1), libreadline5 (>= 5.2), libsnmp15 (>= 5.5), libssl0.9.8 (>=
         0.9.8m-1), logrotate (>= 3.2-11), iproute
PreDepends: adduser
Suggests: snmpd
Conflicts: quagga, zebra, zebra-pj
Replaces: quagga, zebra, zebra-pj
Description: BGP/OSPF/RIP routing daemon
 GNU Quagga is free software which manages TCP/IP based routing protocols. It
 supports BGP4, BGP4+, OSPFv2, OSPFv3, IS-IS, RIPv1, RIPv2, and RIPng as well as
 the IPv6 versions of these.

 As the precessor Zebra has been considered orphaned, the Quagga project has
 been formed by members of the zebra mailing list and the former zebra-pj
 project to continue developing.
```

**Figura 14 – Versión Quagga**

Además, Quagga puede configurase desde un Shell interactivo llamado *vtysh*, el cual ofrece una sintaxis de configuración como la de CISCO.

```
vyatta@vyatta# vtysh

Hello, this is Quagga (version 0.99.16).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

vyatta# configure terminal
vyatta(config)# interface eth1
vyatta(config-if)# ip address 10.3.3.3/24
vyatta(config-if)# _
```

**Figura 15 - vtysh**

Como se aprecia en la figura 15, al ejecutar la instrucción, lo primero que el sistema muestra es la versión de Quagaa. A partir de ahí, se podrá configurar algunas opciones como la dirección IP, los servicios RIP, BGP, OSPF, etc. Para salir ejecutaremos *exit* y se podrá comprobar que los cambios han quedado guardados de la misma forma que si se hubieran efectuado desde el la consola de Vyatta.

## 2.10. Servicios

Acto seguido se procederá a ver como configurar los servicios que nos ofrece Vyatta, como por ejemplo, DNS, SSH, Telnet, etc.

### 2.10.1. Telnet

Mediante el protocolo de Telnet se podrá acceder mediante una red al sistema Vyatta para manejarla remotamente como si se estuviera en ella. Para conectarse a este servicio Vyatta usa el puerto 23 por defecto, aunque se podrá cambiarlo. El mayor problema de este servicio es la seguridad, ya que todos los nombres de usuario y contraseñas necesarias para entrar en el sistema, viajan por la red como texto plano. Por esta razón de seguridad, este servicio ya no es usado apenas, dejando paso a ssh.

Para habilitar el servicio telnet en el sistema se ejecutará:

```
vyatta@R1# set service telnet
vyatta@R1# commit
```

Después de activar el servicio se configurarán algunas opciones, ya que son todas opcionales.

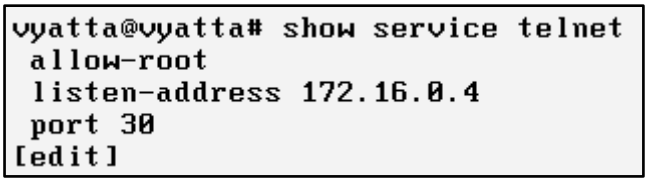


**Figura 16 - Telnet**

En la figura 16 se observan los tres parámetros que se pueden configurar en el servicio telnet.
En primer lugar se permite el acceso mediante telnet al usuario root.

```
vyatta@R1# set service telnet allow-root
vyatta@R1# commit
```

Luego se permite que solo una dirección IP pueda utilizar telnet. Esto puede usarse como medida de seguridad, ya que por defecto cualquier maquina conectada a la red podría hacer una petición telnet. Es una manera de limitar el acceso al sistema.

```
vyatta@R1# set service telnet listen-address 172.16.0.4
vyatta@R1# commit
```

Por último se establece que el puerto por defecto por el que el sistema escuche las peticiones no sea el 23 sino que sea el 30.

```
vyatta@R1# set service telnet port 30
vyatta@R1# commit
```

Con estos sencillos pasos se habrá configurado el servicio telnet según las necesidades. Como se ha comentado, este servicio no es seguro, por lo que si se necesita que la conexión sea lo más segura posible, habrá que habilitar y configurar el servicio ssh.

### 2.10.2. SSH

Mediante el protocolo ssh se podrá acceder al router Vyatta remotamente a través de la red y manejar por completo todas las instrucciones mediante el intérprete de comandos. Todo ello dependerá del usuario con el que se autentique en el sistema. Configurar ssh es opcional, pero es una práctica recomendada para proveer al sistema Vyatta un acceso seguro remoto. Además de la autenticación de contraseña estándar proporcionado por ssh, también está disponible la autenticación con clave pública compartida.

Por defecto Vyatta utiliza el puerto 22 para este servicio aunque se podrá modificar para que utilice el puerto que el administrador quiera.

Para habilitar el servicio ssh en el sistema se escribirá:

```
vyatta@R1# set service ssh
vyatta@R1# commit
```

Después de activar el servicio ssh, se podrá configurar aspectos como el puerto de escucha, las versiones, qué direcciones IP pueden acceder, etc.

```
vyatta@vyatta# show service ssh
 allow-root
 listen-address 172.16.0.1
 listen-address 172.16.0.5
 port 50
 protocol-version all
[edit]
```

**Figura 17 – SSH**

En la figura 17 se observa algunos de los parámetros que se pueden configurar en el servicio ssh. A continuación se enumerarán y se verá cómo activarlos.

En primer lugar se puede permitir, o no, que el usuario root se pueda autenticar.

```
vyatta@R1# set service ssh allow-root
vyatta@R1# commit
```

Otro aspecto a configurar, al igual que en el servicio telnet, puede ser una lista de las direcciones IP que se quiera que puedan acceder por ssh al sistema, ya que por defecto el servicio ssh aceptará cualquier petición IP del sistema.

```
vyatta@R1# set service ssh listen-address 172.16.0.1
vyatta@R1# set service ssh listen-address 172.16.0.5
vyatta@R1# commit
```

Como se ha comentado anteriormente, también se puede cambiar el puerto de escucha del servicio ssh, que por defecto es el 22.

```
vyatta@R1# set service ssh port 50
vyatta@R1# commit
```

También se puede especificar que versión de ssh esté habilitada, la versión 1, la 2 ó ambas. Por defecto viene habilitada la versión 2 ya que es más segura. En este caso se han habilitado ambas.

```
vyatta@R1# set service ssh protocol-version all
vyatta@R1# commit
```

Con estos simples pasos se habrá configurado el servicio ssh, dándole seguridad.

### 2.10.3. DHCP

DHCP es un protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstos van solicitándolas.
Sin DHCP, cada dirección debe configurarse manualmente en cada dispositivo y, si el dispositivo se mueve a otra subred se debe configurar otra dirección IP diferente. El DHCP permite al administrador supervisar y distribuir de forma centralizada las direcciones IP necesarias y, automáticamente, asignar y enviar una nueva IP si el dispositivo se conectara en un lugar diferente de la red.

En primer lugar, y al igual que todos los demás servicios, habrá que activarlo, ya que por defecto Vyatta tiene deshabilitados los servicios.

```
vyatta@R1# set service dhcp-sever
vyatta@R1# commit
```

A continuación habrá que configurar grupos de direcciones para que el sistema actúe como un servidor DHCP para la red.

Se va a crear un escenario sencillo con dos interfaces, cada uno conectado a una red diferente, como se puede observar en la figura 18.

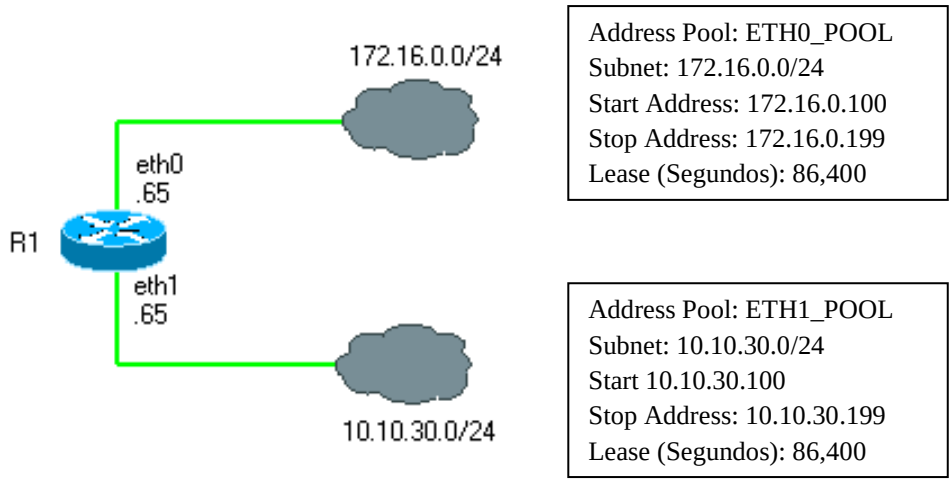


**Figura 18 – DHCP**

Para ello se seguirán los siguientes pasos en modo configuración.
Primero se establecerá un nombre al grupo de direcciones de la red, en qué dirección empezará y en cuál terminará.

***vyatta@R1# set service dhcp-server shared-network-name ETH0_POOL subnet 172.16.0.0./24 start 172.16.0.100 stop 172.16.0.199***

Se pueden especificar diferentes aspectos como el router por defecto,

***vyatta@R1# set service dhcp-server shared-network-name ETH0_POOL subnet 172.16.0.0./24 default-router 172.16.0.65***

la dirección de un servidor DNS,

***vyatta@R1# set service dhcp-server shared-network-name ETH0_POOL subnet 172.16.0.0./24 dns-server 172.16.0.34***

un nombre de dominio,

***vyatta@R1# set service dhcp-server shared-network-name ETH0_POOL subnet 172.16.0.0./24 domain-name grupo1***

o por cuánto tiempo será válida la dirección IP. En este caso se establecerá un valor de 86400 segundos, que es lo mismo que 1 día.

```
vyatta@R1# set service dhcp-server shared-network-name ETH0_POOL
subnet 172.16.0.0./24 lease 86400
vyatta@R1# commit
```

Y exactamente igual para la otra red.
Para ver los cambios que se han realizado, se utiliza la *instrucción* ***show service dhcp-server***, y se pueden ver las distintas opciones que se han ido configurado en forma esquemática.

```
vyatta@R1# show service dhcp-server
    shared-network-name  ETH0_POOL  {
        subnet  172.16.0.0/24  {
            default-router  172.16.0.65
            dns-server  172.16.0.34
            start  172.16.0.100  {
                stop  172.16.0.199
            }
        }
    }
    shared-network-name  ETH1_POOL  {
        subnet  10.10.30.0/24  {
            default-router  10.10.30.65
            dns-server  10.10.30.34
            start  10.10.30.100  {
                10.10.30.199
            }
        }
    }
```

Después de configurar el servidor DHCP, se conecta un PC, en el cual se tiene ejecutado un cliente dhcp, enviando peticiones "broadcast" solicitando información de configuración. Por defecto estas peticiones se realizan contra el puerto UDP 68. El servidor responde a través del puerto UDP 67 proporcionando al cliente una dirección IP, en este caso, le asigna la primera del conjunto que hemos establecido (172.16.0.100) junto con otros parámetros relevantes para el correcto funcionamiento del sistema en la red, tales como la máscara de red, el router por defecto y el servidor DNS. Toda esta información es válida sólo durante un determinado período de tiempo, de esta forma direcciones IP asignadas a clientes que ya no se encuentran conectados a la red pueden ser reutilizadas al pasar determinado periodo de tiempo.

```
▷ Frame 1 (342 bytes on wire, 342 bytes captured)
▷ Ethernet II, Src: QuantaCo_c3:9e:f1 (00:1b:24:c3:9e:f1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Internet Protocol, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
▷ User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
▷ Bootstrap Protocol
```

**Figura 19 – Captura DHCP**

El sistema Vyatta puede actuar también como un **DHCP relay**, o lo que es lo mismo, puede reenviar las solicitudes de DHCP a otro servidor DHCP.

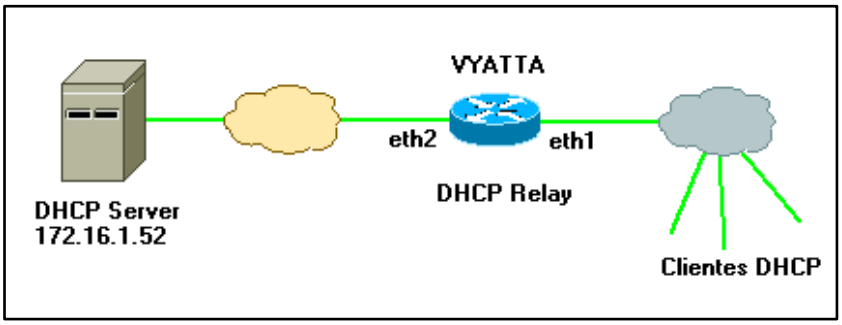


**Ilustración 20 – DHCP Relay**

A continuación se va a mostrar como configurar el escenario de la figura 20 para que el sistema Vyatta actúe como un servidor DHCP relay.

En primer lugar, se habilitarán ambos interfaces para el servicio DHCP. El router estará esperando recibir las peticiones de los clientes, las cuales van dirigidas al servidor DHCP a través del interfaz eth1 Esto, reenviará los mensajes DHCP del cliente al servidor a través del interfaz eth2.

```
vyatta@R1# set service dhcp-relay interface eth1
vyatta@R1# set service dhcp-relay interface eth1
```

Después se especificará la dirección IP del servidor DHCP.

```
vyatta@R1# set service dhcp-relay server 172.16.1.52
```

Por último, se establecerá la regla de descartar los mensajes que contengan información de un paquete relay. Esto lo se hace para que no se permita el reenvío de mensajes DHCP. Si se recibe un paquete que ya contiene información de relay, el paquete es descartado.

```
vyatta@R1# set service dhcp-relay relay-options relay-agents-packets
discard
vyatta@R1# commit
```

Se pueden establecer diferentes opciones adicionales, como el puerto que será usado para el reenvío de los paquetes DHCP a los clientes, tamaño máximo de los paquetes DHCP, y el número máximo de saltos para que el paquete sea descartado.

Al igual que en otros servicios, Vyatta proporciona la funcionalidad de servidor DHCP y de reenvío para IPv6 (DHCPv6).

### 2.10.4. DNS

El sistema de nombres de dominio (DNS), es un servicio de directorio de internet que proporciona asignaciones entre los nombres de dominio legible por las personas y las direcciones IP. Si un dispositivo necesita acceder a un host a través de internet envía una consulta DNS a un servidor de nombres, éste consulta su registro de nombres y devuelve una respuesta con la dirección IP del nombre especificado. Si no encuentra el registro solicitado, consulta en otro servidor de nombres, y si no en otro, y así sucesivamente. El sistema de Vyatta soporta las 3 principales características de DNS:

- Sistema DNS

En un sistema DNS se define la lista de nombres de dominio que el sistema Vyatta puede usar para resolver nombres a direcciones IP. Esta lista se crea utilizando la instrucción ***set system name-server***.

- DNS Dinámico

Esta característica es especialmente útil para sistemas en los que se proporciona una IP dinámica por el proveedor de servicios de internet (ISP). Cada vez que cambia la dirección IP, el sistema Vyatta actualiza el servidor de servicios DNS Dinámico (DDNS) con el cambio. El proveedor de DDNS es responsable de propagar este cambio a otros servidores DNS. El sistema de Vyatta es compatible con varios proveedores de DDNS. En Vyatta se puede realizar esta acción mediante la instrucción ***set service dns dynamic*.**

- Reenvío DNS

Cuando el reenvío de DNS se utiliza, el router del cliente, ofrece su propia dirección IP (que es estática) como la del servidor DNS para los equipos de su red, de modo que todas las peticiones DNS de los clientes se hacen por la dirección del router del cliente.

Cuando se hacen las peticiones DNS, el router del cliente las envía al servidor DNS del ISP; las respuestas se dirigen de nuevo al router del cliente y se reenvían a través de los hosts. Si el ISP cambia la dirección de su servidor DNS, el router del cliente, simplemente registra la nueva dirección del servidor. Para los clientes de la LAN, la dirección del servidor no cambia. Otra ventaja para el reenvío de DNS, es que las peticiones DNS se almacenan en la caché del sistema Vyatta, hasta que el valor del tiempo de vida del registro DNS caduca o se llena la memoria caché.
En Vyatta se puede realizar esta acción mediante la instrucción ***set service dns forwarding*.**

### 2.10.5. Web Caching

El sistema de Vyatta puede ser configurado para actuar como un servidor proxy web para el almacenamiento en caché web y filtrado web. Un cliente puede solicitar una página web desde el sistema de Vyatta, que se conecta al servidor web, y pide la página en nombre del cliente. El sistema Vyatta almacena la respuesta; si la página se solicita de nuevo, se puede tomar directamente de la caché, ahorrando tiempo y el ancho de banda necesario para realizar transacciones con el servidor web.
Por defecto, el sistema actúa como un proxy transparente, el cual redirige automáticamente el tráfico HTTP (puerto 80) al servidor proxy web (que por defecto usa el puerto 3128). Para configurar este servicio se usará la instrucción ***set service webproxy listen-address*** <*ipv4*>**.**
También se puede configurar el sistema de Vyatta como un proxy no transparente usando la instrucción ***set service webproxy listen-address*** <*ipv4*> ***disable-transparent.***

Después de activar el web caching, se pueden configurar algunos aspectos como el tamaño del servidor proxy cache, el puerto por defecto por el que escuchará las peticiones, etc.

### 2.10.6. LLDP

El Link Layer Discovery Protocol (LLDP) es un protocolo de descubrimiento de "vecinos" estándar, alternativo a Cisco Discovery Protocol (CDP) que proporciona información acerca del switch al que estamos conectados. La información recogida con LLDP sobre dispositivos de red como conmutadores y puntos de acceso inalámbricos, ayuda a localizar problemas y permite a los sistemas de gestión crear visualizaciones precisas de la topología de la red.

Para utilizar LLDP en el sistema Vyatta, al igual que con la mayoría de los servicios, habrá que habilitarlo, en el modo configuración, mediante la instrucción ***set service lldp.***

Una vez que el servicio está habilitado, se puede registrar la información sobre la ubicación del dispositivo, la dirección y el puerto de gestión, así como los protocolos de herencia que soporta. Toda la información adicional, incluyendo las capacidades de configuración del sistema y vecinos, se extrae automáticamente del sistema y se almacena en una base de datos MIB (Base de Información Gestionada), la cual es accesible vía SNMP. Una MIB, es un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, se todos los dispositivos gestionados en una red de comunicaciones.

### 2.10.7. SNMP

El SNMP es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Este mecanismo, permite a los administradores supervisar el funcionamiento de la red, buscar y resolver sus problemas, y planear su crecimiento. El sistema Vyatta soporta SNMP a través de redes IPv4 e IPv6.

El SNMP utiliza un modelo de gestor/agente para la administración de los dispositivos. El agente reside en el dispositivo, y proporciona la interfaz para el dispositivo físico que se administra. El gerente reside en el sistema de gestión y proporciona la interfaz entre el usuario y el agente SNMP. La interfaz entre el gestor SNMP y el agente SNMP utiliza una base de administración (MIB) y un pequeño conjunto de comandos para el intercambio de información.

Para configurar SNMP, debe haber al menos un usuario creado, y el modelo MIB que debe cargarse.

### 2.10.8. QoS

Calidad de Servicio (QoS) es una característica que permite a los administradores de red identificar diferentes flujos de tráfico y luego tratarlos de acuerdo a sus requerimientos individuales, en lugar de simplemente utilizar un mecanismo por defecto, el cual no hace diferenciación del tráfico.

En el sistema de Vyatta, el mecanismo QoS se basa en la priorización de la cola. Además de los mecanismos de colas, el sistema ofrece una gran variedad de mecanismos para identificar y tratar los distintos flujos de tráfico que pasan a través de un interfaz. En general, estos se pueden clasificar como los mecanismos que se aplican para el tráfico saliente y el entrante.

El flujo de trabajo general para los mecanismos de QoS no predeterminados (políticas de tráfico) se establece de la siguiente manera:

- Crear una política de tráfico. Ésta, identifica los flujos de tráfico y especifica cómo se va a tratar cada flujo.
- Aplicar la política a un interfaz.

## 2.11. Routing

Vyatta nos ofrece distintas opciones de enrutamiento, dependiendo de las necesidades y las dimensiones de nuestra red.

### 2.11.1. Rutas Estáticas

Una ruta estática es una ruta configurada manualmente, por lo que no se podrá actualizar dinámicamente en el caso de que se añadan o se modifiquen elementos de la red.

La colección de todas las rutas que el router Vyatta ha aprendido, bien hayan sido introducidas manualmente o hayan sido aprendido dinámicamente, son almacenadas en su base de información de enrutamiento (RIB).

Si por lo que sea un enlace falla, el router elimina de su base de información de enrutamiento las rutas estáticas, que utilizan esta interfaz para llegar al siguiente salto. En general, las rutas estáticas se suelen utilizar para topologías de red muy simples, o para reemplazar el comportamiento de un protocolo de enrutamiento dinámico para un número pequeño de rutas.

Mediante la instrucción ***set prototcols static route*** *<red>* ***next-hop*** ***<****IPv4****>*** se creará una ruta estática en la que llegaremos a la “red” mediante la “IP” de salto.

### 2.11.2. RIP

Routing Information Protocolo (RIP) es un protocolo de enrutamiento dinámico adecuado para redes homogéneas pequeñas. Está clasificado como protocolo IGP y emplea el algoritmo de enrutamiento de vector-distancia. RIP determina el mejor camino al contar los saltos hasta el destino. El número máximo de saltos es 15 (16 ya es considerado distancia infinita).

Mediante la instrucción ***set protocols rip*** se habilitará este servicio, y después mediante las diferentes opciones se establecerá la configuración que se adecue a la red.

Vyatta soporta las versiones RIPv1 y RIPv2, aunque no hay manera de activarlas por separado cómo es posible en CISCO.

### 2.11.3. OSPF

Open Shortest Path First (OSPF) es un protocolo dinámico de enrutamiento que usa el algoritmo de Dijkstra, al contrario de otros protocolos (como RIP) que usan el algoritmo vector distancia, para calcular la ruta más corta posible.

En OSPF, cada router anuncia el estado de sus propias conexiones en un anuncio de estado de enlace (LSA) mediante una multidifusión a otros routers de la red. Además, cada router que usa LSA recibe información para la construcción de un gráfico que representa la topología de la red.

OSPF es probablemente el tipo de protocolo IGP más utilizado en redes grandes, siendo el sucesor de RIP.

Para configurar OSPF, primero se definirá una IP que identificará al router, ya que sino le será asignada una por defecto. Después, al igual que RIP, se anunciarán las redes a las que está conectado con otro router mediante el comando ***set protocols ospf area 0.0.0.0 network*** *<IP/máscara>*. Si está conectado a una red en la que no hay mas routers conectado entones se usará la instrucción ***set protocols ospf redistribute connected***.

Ésta sería una configuración básica, en la que después se podrían configurar aspectos como la distancia para las rutas OSPF, definir un vecino OSPF directamente, definir interfaces del router como pasivos (podrá recibir tráfico por ellos pero no podrán enviar nada para actualizaciones), etc.

### 2.11.4. BGP

Border Gateway Protocol (BGP) es un protocolo mediante el cual se intercambia información de encaminamiento entre sistemas autónomos. Por ejemplo, los ISP registrados en internet suelen componerse de varios sistemas autónomos y para este caso es necesario un protocolo como BGP.

El principal concepto de BGP es el Sistema Autónomo (AS) que permite configurar y delimitar la información que contiene. Cada AS tendrá sesiones internas (iBGP) y sesiones externas (eBGP).

En el sistema Vyatta se puede configurar iBGP, eBGP y múltiples características de éstos. Para más información consultar manual.

## 2.12. Seguridad

Además de todos los protocolos de enrutamiento y servicios vistos anteriormente, el sistema de Vyatta incorpora unos mecanismos de seguridad que terminan de completar las características de este enrutador.

### 2.12.1. Firewall

La función del firewall es filtrar y analizar los paquetes IP entre los interfaces de las redes. El uso principal es para proteger el tráfico entre la red interna e Internet. Esto permite filtrar los paquetes de unas determinadas características y realizar acciones en los paquetes que coincidan con las reglas que queramos.

Las diferentes funciones del firewall de Vyatta son el filtrado de paquetes que atraviesan el router, de la red local a Internet, filtrado de paquetes en la red local, reglas que incluyen dirección IP origen, destino, puertos y paquetes ICMP.

El sistema de Vyatta también ofrece protección para entornos IPv6, ya que en éstos no disponen de la funcionalidad NAT. Además, un firewall es la única manera de proteger una red IPv6. Hay que decir que las reglas para IPv4 son completamente independientes de las reglas para IPv6.

Mediante la instrucción ***set firewall*** se habilitará el firewall para IPv4, para después, mediante las diferentes opciones de este comando definir las reglas que se necesiten para proteger nuestra red. Para IPv6 se añadirá ***ipv6***. Para ver las reglas definidas se usará ***show firewall***.

### 2.12.2. VPN

Una Red Privada Virtual (VPN) es una tecnología de red que permite crear una extensión de la red local sobre una red pública o no controlada, como por ejemplo Internet. Gracias a una VPN se puede conectar dos o más redes de una empresa utilizando como vínculo Internet o permitir a un usuario acceder a su equipo desde un sitio remoto, como por ejemplo un hotel. Todo ello utilizando la infraestructura de Internet. El sistema Vyatta soporta dos tipos de modos VPN:

- Punto a punto, en el que VPN permite conectar dos o más sitios separados una gran distancia de tal manera que parecen estar en una única red privada. El sistema Vyatta utiliza en este modo, el protocolo de comunicaciones IPsec y OpenVPN.

- De acceso remoto, en el que se crea un túnel VPN que se establece entre un usuario remoto y un servidor VPN. Esto permite a un empleado acceder a la red privada de la empresa desde su propia casa. El sistema Vyatta utiliza en este modo, el protocolo de comunicaciones PPTP, L2TP, IPsec y OpenVPN.

Conceptualmente, los dos tipos de VPN son bastantes similares, en el sentido de que ambos usan un "túnel" para que los dos extremos crean que están en la misma red, pero las soluciones varían en la forma en la que se establece el túnel.

### 2.12.3. IPS

Un Sistema de Prevención de Intrusos (IPS) es un dispositivo que ejerce el control de acceso en una red informática para proteger a los sistemas computacionales de ataques y abusos. Cuando un ataque es detectado, el sistema puede descartar los paquetes maliciosos y mientras permitir que todo el tráfico restante pase.

El sistema Vyatta utiliza el motor Snort (www.snort.org) para la detección y prevención de intrusiones. Snort puede ser utilizado para detectar una gran variedad de ataques y sondas, tales como desbordamientos de búffer, escaneos de puertos, ataques CGI, sondas SMB, y muchos más. Snort utiliza un lenguaje de reglas flexible para describir el tráfico que debe recolectar, así como un motor de detección que utiliza un plug-in de la arquitectura modular.

Vyatta ofrece una versión gratuita del IPS para el modelo Vyatta Core, y otra basada en una subscripción para el Vyatta PLUS. En la versión de pago, los subscriptores reciben las reglas nuevas a medida que las nuevas vulnerabilidades son descubiertas.

Antes de utilizar el IPS, habrá que registrarse en la página de snort, y así obtener un código. Este código se utilizará para descargar las reglas de snort y posteriormente se establecerá con qué frecuencia se quiere que éstas sean actualizadas. Después se especificará el tráfico a inspeccionar debido a su contenido malicioso. Por último se definirá cómo tratar ese tráfico malicioso.

### 2.12.4. Web Filtering

Como se dijo en el apartado de Web Caching, el sistema de Vyatta puede ser configurado para actuar como un servidor proxy web para el almacenamiento en caché web y filtrado web. El filtrado web (también llamado filtrado URL) es una herramienta importante para la gestión de acceso web, que reduce la exposición y amenazas basadas en web, limita el bloqueo de contenido ofensivo, aumenta la productividad, y administra el ancho de banda.

## 2.13. Acceso Remoto [16]

El sistema Vyatta puede ser habilitado para la ejecución remota de comandos en la versión VSE a través de HTTPS usando un sencillo interfaz. Además de poder acceder al conjunto estándar de comandos operativos y de configuración, esta API proporciona el control de procesos y funciones de gestión. La API se basa en el principio de REST, y usa el formato JSON para la representación de datos.

Los requisitos para su funcionamiento son:

- Disponer de la versión VSE de pago de Vyatta con el servicio HTTPS habilitado
- Ser un usuario del sistema Vyatta
- Tener un sistema que genera las peticiones HTTPS de Vyatta.

# 3.- PRÁCTICAS

A continuación se mostrará cómo se puede sustituir los routers CISCO en las prácticas de la asignatura Laboratorio de Programación de Redes (LPR), por el sistema open-source Vyatta, en concreto la versión VC6.2. Se verá que las instrucciones son, en la mayoría de los casos, similares y que la adaptación al software Vyatta para un usuario de CISCO no resulta difícil.

## 3.1. Práctica 1 - Conexión de PCs a través de modem

En está practica se ve cómo utilizar los puertos serie de los PCs en Linux. Se usarán para establecer un enlace entre dos PCs simulando que éstos están en ubicaciones alejadas, por lo que no se podrán conectar con un simple cable serie. Para ello se colocarán unos módems que conviertan las señales de los puertos serie en señales que se puedan transmitir por una línea telefónica tradicional.
Al no utilizar routers CISCO para la comparación con el sistema Vyatta, no está dentro de los objetivos del proyecto.

## 3.2. Práctica 2 - PCs en redes de área local Ethernet

Para probar las configuraciones de redes empleando routers se necesitarán PCs, los cuales se colocarán en las diferentes redes. Por ello en esta práctica se repasan los comandos y ficheros básicos para configurar un interfaz de red Ethernet con IP en Linux y conectarlo a una LAN con un router de acceso. Para ello se emplearán diferentes elementos típicos de redes Ethernet como los hubs y los switches.
Al no utilizar routers CISCO para la comparación con el sistema Vyatta, no está dentro de los objetivos del proyecto.

## 3.3. Practica 3 - Configuración IP de interfaces Ethernet en CISCO IOS

En esta práctica se ve cómo configurar desde un nivel básico los interfaces ethernet en Cisco, por lo que se hará una comparación con el sistema Vyatta.

Los dispositivos Cisco poseen su propio sistema operativo, Cisco IOS (Internetworking Operating System). El Cisco IOS CLI (Command-Line Interface) es el método principal para la configuración, monitorización y mantenimiento de equipos Cisco. El acceso a este interfaz se realiza normalmente a través del puerto de consola de los routers. El interfaz CLI permite ejecutar comandos del Cisco IOS y una vez configurado el router

se puede permitir el acceso a dicho interfaz a través de un interfaz de red además de por el puerto de consola.

El software de Vyatta es en cambio una distribución Linux basada en Debian, diseñado para transformar el hardware estándar x86 en un router/firewall. Vyatta posee también su propio sistema operativo (Vyatta Network OS), el cual se configura al igual que Cisco mediante un interfaz de línea de comandos, o mediante un interfaz gráfico.

La primera configuración del router CISCO es necesario hacerla a través de un puerto de consola, ya que al no tener el router aún una configuración de red no se puede acceder a él por ninguno de sus interfaces de red.

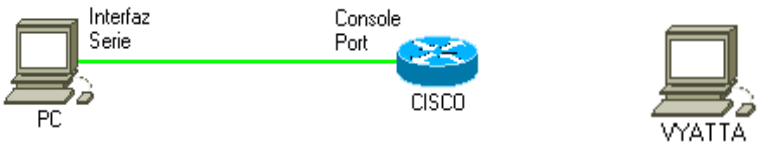


**Figura 21 – Acceso Cisco y Vyatta**

En cambio con el software Vyatta no se necesita más que el PC donde esté instalado ya que será éste el que actúe como router.

En los dos casos se puede ver la versión que tenemos mediante el comando ***show version***.

```
vyatta@vyatta:~$ show version
Version:      VC6.2-2011.02.09
Description:  Vyatta Core 6.2 2011.02.09
Copyright:    2006-2011 Vyatta, Inc.
Built by:     autobuild@vyatta.com
Built on:     Wed Feb  9 20:04:26 UTC 2011
Build ID:     1102092009-7353197
Boot via:     disk
Uptime:       08:39:06 up 59 min,  1 user,  load average: 0.00, 0.00, 0.00
```

**Figura 22 – Versión de Vyatta**

Como se observa en la figura 22, es la última versión del software de Vyatta, en concreto la VC6.2, que salió el 2-9-2011. También se ve reflejado de qué modo se está arrancando el software, en este caso se ve que pone "*boot via: disk*", lo que significa que se está arrancando el sistema Vyatta desde un disco duro, en el que ha sido instalado previamente.

### 3.3.1. Configuración IP básica de un interfaz Ethernet en Vyatta.

Al arrancar Vyatta e introducir las credenciales, se muestra lo siguiente:

```
vyatta@vyatta:~$
```

Es el modo operacional, con el que se pueden realizar tareas como mostrar información, reiniciar y apagar el sistema, etc.
El siguiente paso que habrá que mirar será ver si el equipo actúa como router. Aunque esta opción viene por defecto activada, se puede comprobar que realmente lo está con la siguiente instrucción:

```
vyatta@vyatta:~$ show ip forwarding
```

A lo que el sistema responde "***IP forwarding is on".***

Como se comentó en la parte del manual, para realizar cualquier cambio en la configuración del router Vyatta, hay que entrar en modo configuración mediante el comando:

```
vyatta@vyatta:~$ configure
```

Al realizar esa acción, ahora se verá el prompt así:

```
vyatta@vyatta#
```

A continuación, se va a configurar la dirección IP de uno de los interfaces Ethernet. Al contrario que en CISCO en el que se tenía que pasar al modo de configuración del interfaz que se quería modificar y después activarlo, en Vyatta, solamente mediante un comando se realizará la configuración.

```
vyatta@vyatta# set interfaces ethernet ethx address IP/netmask
```

En donde ethx corresponde al interfaz que se quiera modificar, IP a la dirección de red y netmask a la máscara de red en formato CIDR.
Para realizar una prueba, se va a realizar el escenario de la figura 23.

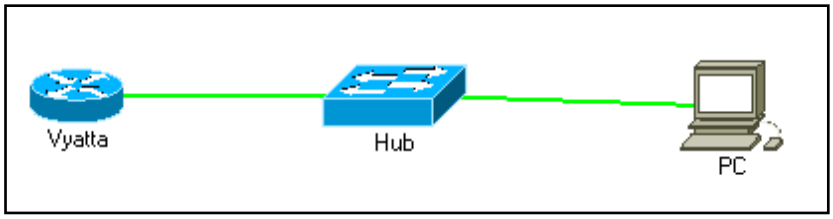


**Figura 23 – Interfaces**

Como está basado en GNU/Linux, la manera de nombrar las interfaces de red es mediante ethX, con X igual a 0 para la primera interfaz que reconoce el sistema, 1 la segunda, etc. Para saber las interfaces que tiene el router, se ejecutará el comando:

***vyatta@vyatta:~$ show interfaces***

| *Interface* | *IP Address* | *State* | *Link* | *Description* |
|---|---|---|---|---|
| *eth0* | - | *up* | *down* | |
| *eth1* | - | *up* | *down* | |
| *eth2* | - | *up* | *down* | |
| *eth3* | - | *up* | *down* | |
| *eth4* | - | *up* | *down* | |
| *lo* | *127.0.0.1/8* | *up* | *up* | |

Este comando es muy útil, ya que gracias a él se verá el estado de los interfaces, si tienen asignada una dirección IP y alguna descripción.

El siguiente paso es configurar las interfaces para asignarles una IP. Por ejemplo, para asignar la dirección 10.1.1.1 a la interfaz eth1:

***vyatta@vyatta# set interfaces ethernet*** *eth1* ***address*** *10.1.1.1/24*

Se puede configurar distintos parámetros como el nombre del host, el nombre de dominio, etc.
En este punto cabe destacar, que en el modo configuración, la instrucción ***set*** sirve para marcar cambios, pero éstos no se hacen efectivos hasta que no se confirmen con ***commit***. Esto se hace como medida de seguridad. Con la instrucción ***show*** se podrán ver los cambios que se hayan realizado, y aquellos que no estén confirmados aparecerán con un símbolo '+' por delante.

Para comprobar todos los parámetros que se hayan configurado, se puede utilizar la instrucción ***show***. En concreto si se quiere ver la configuración de los interfaces de red:

```
vyatta@R1# show interfaces
  ethernet eth0 {
        address 10.1.1.1/24
        hw-id 00:0c:29:af:c9:4b
  }
```

Como se ve, esta instrucción mostraría todas las características de las interfaces que se han configurado. En este caso sólo muestra el interfaz ethernet eth0, ya que no se ha configurado ningún otro.

### 3.3.2. Configuración IP básica del router Vyatta.

Ahora se va a configurar el router para que esté conectado a una red diferente con cada uno de sus interfaces Ethernet. En la figura 24 se ve la topología que se quiere conseguir, donde por simplicidad se han enumerado los interfaces Ethernet del router.

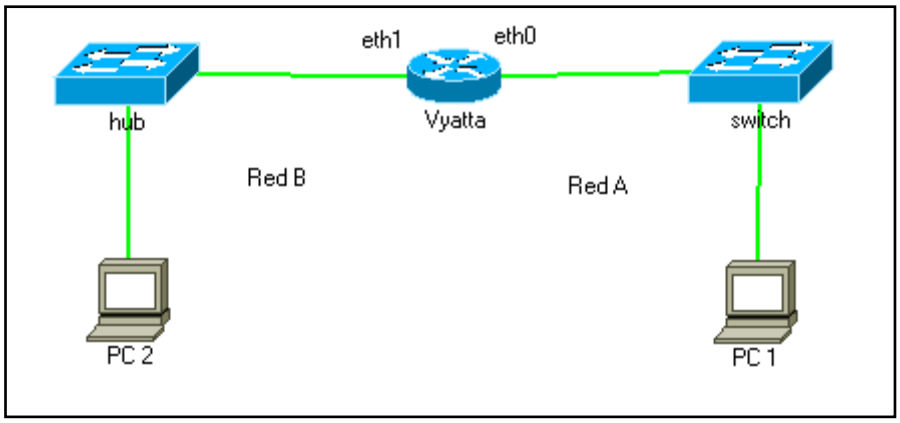


**Figura 24 – Router dos redes**

Se dispone de la red 10.1.1.0/24 que se divide en dos subredes, por un lado la Red A (10.1.1.0/25) y por otro lado la Red B (10.1.1.128/25).

En el router Vyatta solamente habrá que ejecutar las siguientes instrucciones:

```
vyatta@vyatta:~$ configure
vyatta@vyatta# set interfaces ethernet eth0 address 10.1.1.1/25
vyatta@vyatta# set interfaces ethernet eth1 address 10.1.1.129/25
vyatta@vyatta# commit
```

En los PCs también habrá que configurar una dirección IP y la puerta de enlace por defecto.

***usuario@PC1:~$ sudo ifconfig*** *eth0 10.1.1.2* ***netmask*** *255.255.255.128*
***usuario@PC1:~$ sudo route add defautlt gw*** *10.1.1.1*

***usuario@PC2:~$ sudo ifconfig*** *eth0 10.1.1.130* ***netmask*** *255.255.255.128*
***usuario@PC2:~$ sudo route add defautlt gw*** *10.1.1.129*

Con estos sencillos pasos se habría configurado el escenario, en el que se puede ver como con dos simples instrucciones quedarían configurados los interfaces Ethernet del router Vyatta.

### 3.3.3. Topología con dos routers Vyatta.

A continuación se va a crear la topología de figura 25 para probar el funcionamiento de dos routers Vyatta en 3 redes diferentes.

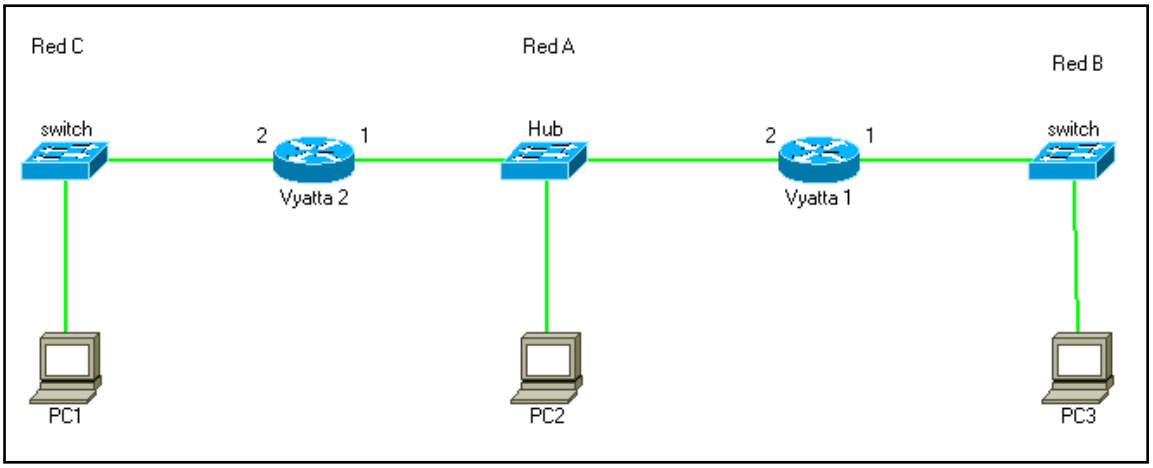


**Figura 25 – Dos routers**

Se dispone de la red 10.3.3.0/24 que se dividirá en 3 subredes, Red A (10.3.3.64/26), Red B (10.3.3.128/26) y Red C (10.3.3.192/26).

En el router Vyatta 1 se ejecutarán las siguientes instrucciones para la configuración de los interfaces:

***vyatta@vyatta1:~$ configure***
***vyatta@vyatta1# set interfaces ethernet*** *eth1* ***address*** *10.3.3.129/26*
***vyatta@vyatta1# set interfaces ethernet*** *eth2* ***address*** *10.3.3.65/26*
***vyatta@vyatta1# commit***

En el router Vyatta 2 se ejecutarán las siguientes instrucciones para la configuración de los interfaces:

```
vyatta@vyatta2:~$ configure
vyatta@vyatta2# set interfaces ethernet eth1 address 10.3.3.66/26
vyatta@vyatta2# set interfaces ethernet eth2 address 10.3.3.193/26
vyatta@vyatta2# commit
```

Y para la configuración de las rutas estáticas:

```
vyatta@vyatta1# set protocols static route 10.3.3.192/26 next-hop 10.3.3.66
vyatta@vyatta1# commit
```

Con esto, se indica al router Vyatta1 que si recibe un paquete que va a la red 192.168.3.0/24 (Red C), que lo envíe a la dirección 192.168.1.3.

Al igual que en router 1, se indica al router Vyatta 2 qué camino escoger cuando recibe un paquete que va hacia la red 192.168.2.0/24 (red B).

```
vyatta@vyatta2# set protocols static route 10.3.3.128/26 next-hop 10.3.3.65
vyatta@vyatta2# commit
```

En los PCs también habrá que darles una dirección IP y la puerta de enlace por defecto.

```
usuario@PC1:~$ sudo ifconfig eth0 10.3.3.194 netmask 255.255.255.192
usuario@PC1:~$ sudo route add defautlt gw 10.3.3.193

usuario@PC2:~$ sudo ifconfig eth0 10.3.3.67 netmask 255.255.255.192
usuario@PC2:~$ sudo route add defautlt gw 10.3.3.65

usuario@PC3:~$ sudo ifconfig eth0 10.3.3.130 netmask 255.255.255.192
usuario@PC3:~$ sudo route add defautlt gw 10.3.3.129
```

Al ejecutar todas estas instrucciones se comprueba que cada PC puede comunicarse con cualquier PC de todas las redes.

Para ver las configuraciones que se han realizado, se pueden ejecutar las siguientes instrucciones:

```
vyatta@vyatta1:~$ configure
vyatta@vyatta1# show protocols
  static {
    route 10.3.3.192 {
        next-hop 10.3.3.66{
        }
    }
  }
```

Aquí se puede ver que en router Vyatta1, hay dos interfaces conectados, cada uno a una red diferente, y también, una ruta estática para cuando reciba un paquete para ella, sepa por cual interfaz enviarla para que llegue a su correcto destino. Ésta última regla se puede ver con mayor claridad mediante:

```
vyatta@vyatta1:~$ show ip route
C>* 127.0.0.0/8 is directly connected, lo
S>* 10.3.3.192 [1/0] via 10.3.3.66, eth2
C>* 10.3.3.64/26 is directly connected, eth2
C>* 10.3.3.128/26 is directly connected, eth1
```

### 3.3.4. Proxy-ARP

Es un mecanismo que permite tener niveles de enlace separados por un router que ejecute Proxy ARP. Su objetivo es simular que todos esos enlaces pertenecen a una única red. Para ello, el router responderá a paquetes ARP dirigidos a destinos situados al otro lado del router, simulando así que dichos destinos son accesibles directamente. En definitiva, el Proxy ARP permite a un interfaz Ethernet responder con su propia MAC a consultas ARP, las cuales no hacen referencia a la dirección IP configurada en ese interfaz.

Para probar esto, partiendo de la figura 25, se cambiará la configuración del PC3, para que su máscara de red haga referencia a todo el espacio de direcciones (es decir, máscara de 24 bits).

A diferencia de los routers CISCO, vyatta tiene deshabilitado en sus interfaces la opción de proxy-arp, por lo que habrá que activarlo en las interfaces en las que se quiera que esté activo.

| ***vyatta@vyatta1# set interfaces ethernet** eth1 **ip enable-proxy-arp*** |
|---|

Como se ve, se ha activado el interfaz Ethernet eth1 del router Vyatta, para que al comunicarse con el PC2, parezca que está en su misma red, aunque sea este interfaz el que diga que es él el PC2.
Para que se pueda ver este proceso, antes se han borrado las entradas de la cache ARP del PC3.

## 3.4. Práctica 4 - PC como router IP

En esta práctica se va a ver cómo es posible que un router Vyatta se pueda comunicar con un PC con Linux, el cual realiza la función de un router.

Se va a emplear un PC como router utilizando el sistema operativo Linux.
Para que funcione como un router hay que convencerle de que cuando reciba un paquete con esas características no lo tire, sino que lo reenvíe aplicando las reglas que tiene en su tabla de rutas. Esta funcionalidad es lo que se conoce como IP forwarding. Para activarlo habrá que escribir un 1 en el fichero ***/proc/sys/net/ipv4/ip_forward***. Con sólo activar el IP forwarding, el PC empezará a reenviar paquetes.

### 3.4.1. Topología con 2 routers, mezclando Vyatta y Linux.

A continuación se va a crear la topología que se ve en la figura 26.

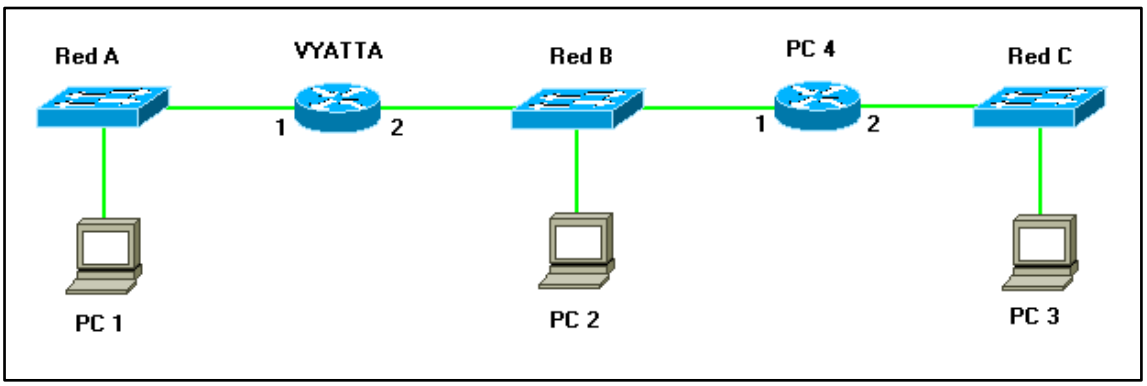


**Figura 26 – Vyatta y Linux**

Como se ha comentado anteriormente, se dispone de un PC con el sistema Vyatta y de otro que solo tiene una distribución de Linux que actuará como router, habiéndolo configurado previamente.

Se dispone de la red 10.3.3.0/24 que se divide en 3 subredes, Red A (10.3.3.64/26), Red B (10.3.3.128/26) y Red C (10.3.3.192/26).

Primero se configuran los interfaces Ethernet del router Vyatta y del router PC 4.

```
vyatta@vyatta1:~$ configure
vyatta@vyatta1# set interfaces ethernet eth1 address 10.3.3.65/26
vyatta@vyatta1# set interfaces ethernet eth2 address 10.3.3.129/26
vyatta@vyatta1# commit

usuario@PC4:~$ sudo ifconfig eth1 10.3.3.130 netmask 255.255.255.192
usuario@PC4:~$ sudo ifconfig eth2 10.3.3.193 netmask 255.255.255.192
```

Despues se procede a configurar los PCs con sus rutas por defecto:

```
usuario@PC1:~$ sudo ifconfig eth0 10.3.3.66 netmask 255.255.255.192
usuario@PC1:~$ sudo route add defautlt gw 10.3.3.65

usuario@PC2:~$ sudo ifconfig eth0 10.3.3.131 netmask 255.255.255.192
usuario@PC2:~$ sudo route add defautlt gw 10.3.3.130

usuario@PC3:~$ sudo ifconfig eth0 10.3.3.194 netmask 255.255.255.192
usuario@PC3:~$ sudo route add defautlt gw 10.3.3.193
```

Por último se establecen las rutas por defecto en ambos routers:

```
vyatta@vyatta1# set system gateway-address 10.3.3.130
vyatta@vyatta1# commit

usuario@PC4:~$ sudo route add defautlt gw 10.3.3.129
```

Se ha establecido una ruta por defecto en Vyatta en vez de una ruta estática, porque al tener solo una red no conectada directamente, todo el tráfico que no sea para una red conectada, será para ella.

## 3.5. Práctica 5 - Configuración de interfaces Serie en Cisco IOS

En ésta práctica se ve cómo es posible conectar routers CISCO mediante el interfaz WAN de éstos. Los routers se pueden emplear para conectar tanto LANs como WANs, por lo que suelen tener disponibles interfaces de ambos tipos. Una WAN (red de área extensa) es un tipo de red de computadores capaz de cubrir grandes distancias, proveyendo de servicio a un país o continente. La más grande y conocida red WAN es Internet.

Dos PCs del laboratorio, en los cuales se está probando el sistema Vyatta, disponen de puertos serie USB conectados a otro PC mediante un cable "USB-USB NULL modem". Por medio de éstos, se podría simular el funcionamiento de los interfaces WAN de Cisco, pero como se comentó en la Parte 1, la versión del software de Vyatta que se está utilizando para este proyecto, no permite la configuración de este tipo de interfaces.
Debido a esto, no se verá las instrucciones necesarias para configurar los interfaces serie en el sistema Vyatta. Para consultar toda la información acerca de estos interfaces consultar el manual [17].

Aunque por otro lado sí que es posible configurar IP empleando ppp sobre esta línea serie, debido a que el sistema Vyatta se basa en una distribución de Linux, en concreto en Debian, y contiene el servicio *pppd*.

Para comprobar su funcionamiento se creará el escenario de la figura 27. Se dispone de la red 10.3.3.0/24 que se divide en 3 subredes, Red A (10.3.3.64/26), Red B (10.3.3.128/26) y Red C (10.3.3.192/26).

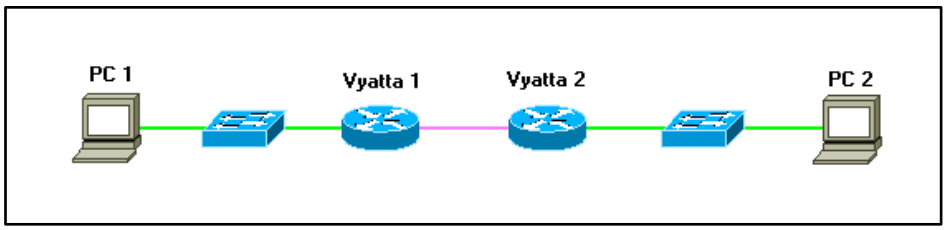


**Figura 27 – Topología en serie**

Primero se establecerá el enlace serie ppp entre los dos routers mediante las siguientes instrucciones:

***vyatta@vyatta1:~$ sudo pppd /dev/ttyUSB0 local nodetach debug***
*10.3.3.129:10.3.3.130*
***vyatta@vyatta2:~$ sudo pppd /dev/ttyUSB0 local nodetach debug***
*10.3.3.130:10.3.3.129*

Como se ve, se especifica en las instrucciones el puerto USB ttyUSB0 y las opciones de *local*, que sirve para indicarle al router que en la línea serie no hay un modem, *nodetach* para que al hacer pppd siga usando el terminal y no se quede corriendo como demonio, *debug* para que muestre información sobre los pasos de establecimiento del enlace con el otro extremo y por último *IPLOCAL:IPREMOTA*, que son las direcciones que se desean emplear en ambos extremos.

Ahora, después de establecer el enlace se puede ver, por ejemplo en el router Vyatta1, que se tiene un nuevo interfaz.

***vyatta@vyatta:~$ show interfaces***

| *Interface* | *IP Address* | *State* | *Link* | *Description* |
|---|---|---|---|---|
| *eth0* | *-* | *up* | *up* | |
| *eth1* | *192.168.1.1/24* | *up* | *up* | |
| *eth2* | *-* | *admin down* | *down* | |
| *eth3* | *-* | *admin down* | *down* | |
| *eth4* | *-* | *admin down* | *down* | |
| *lo* | *127.0.0.1/8* | *up* | *up* | |
| *lo* | *::1/128* | *up* | *up* | |
| ***ppp0*** | ***10.3.3.129*** | ***up*** | ***up*** | |

A continuación, se establecerían las direcciones IP y las rutas por defecto en cada PC, y en los routes también.

Para que si se reinicia el sistema la configuración vuelva a estar, se podrá crear un script, el cual realice los pasos indicados anteriormente, para no tener que ejecutar los mismos comandos cada vez que se reinicie o apague el sistema. Esto no entraría en conflicto con ningún fichero de arranque del sistema, ya que el servicio pppd está instalado para ello en la versión gratuita, no teniendo acceso a los comandos de configuración de las interfaces serie.

## 3.6. Práctica 6 - Configuración de enlaces por interfaces serie en PCs

En esta práctica se ve como emplear IP sobre un enlace entre los puertos serie de los PCs. En concreto a establecer un enlace PPP empleando diferentes velocidades. El protocolo PPP sirve para encapsular paquetes para posteriormente enviarlos por líneas serie.
El sistema Vyatta tiene la opción de configurar el protocolo de encapsulamiento PPP, definiendo el tipo de autenticación que queramos (pap, chap, …), pero el objetivo de esta práctica no es conectar dos routers con este protocolo, sino dos PCs, por lo que no

está dentro de los objetivos del proyecto. Además en la práctica anterior se comprobó que al ser Vyatta una distribución de Linux, es posible configurar dicho protocolo.

## 3.7. Práctica 7 - Rutas estáticas

A continuación se verá cómo introducir manualmente entradas a redes remotas en la tabla de rutas de un router Vyatta.
Como se ha comentado en la Parte 2, las rutas estáticas son una buena elección cuando se tiene que configurar unas redes pequeñas en las que no se van a producir muchos cambios, ya que si no, se usarían los protocolos de enrutamiento RIP u OSPF.

### 3.7.1. Topología en árbol

Se va a reproducir el escenario de la figura 28 para ver los pasos que habría que seguir a la hora de realizar la configuración tanto en el sistema Vyatta como en los PCs.
Se asume que los PCs donde está funcionando el sistema Vyatta disponen de 4 interfaces ethernet.

En la práctica de LPR, la conexión entre los routers CISCO, se hacía mediante interfaces serie. No se ha podido realizar la conexión por puertos serie, no porque no dispongan los ordenadores del laboratorio, sino porque la versión gratuita de Vyatta, aunque reconoce los interfaces, no ofrece la posibilidad de configurarlos. Por lo que se han tenido que unir los routers con un cable cruzado, aunque se podrían haber configurado mediante enlace serie ppp como se explica en la práctica 5.

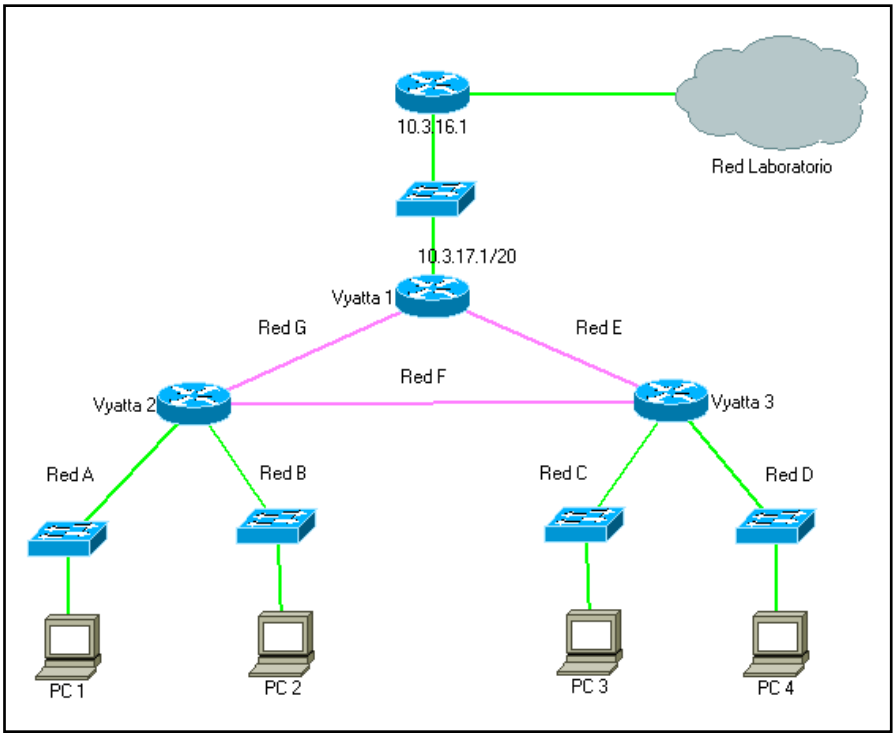


**Figura 28 – Rutas Estáticas**

Para este ejemplo se va a usar la red 10.3.34.0/24 y se va a dividir en 7 subredes (Red A, Red B, …). Por lo que la asignación quedaría así:

Red A → 000XXXXX → 10.3.34.0/27
Red B → 001XXXXX → 10.3.34.32/27
Red C → 010XXXXX → 10.3.34.64/27
Red D → 011XXXXX → 10.3.34.96/27
Red E → 100XXXXX → 10.3.34.128/27
Red F → 101XXXXX → 10.3.34.160/27
Red G → 110XXXXX → 10.3.34.192/27

En el router Vyatta 1 se ejecutarán las siguientes instrucciones para la configuración de los interfaces:

```
vyatta@vyatta1:~$ configure
vyatta@vyatta1# set interfaces ethernet eth1 address 10.3.17.1/20
vyatta@vyatta1# set interfaces ethernet eth2 address 10.3.34.130/27
vyatta@vyatta1# set interfaces ethernet eth3 address 10.3.34.194/27
vyatta@vyatta1# commit
```

En el router Vyatta 2 se ejecutarán las siguientes instrucciones para la configuración de los interfaces:

```
vyatta@vyatta2:~$ configure
vyatta@vyatta2# set interfaces ethernet eth1 address 10.3.34.193/27
vyatta@vyatta2# set interfaces ethernet eth2 address 10.3.34.162/27
vyatta@vyatta2# set interfaces ethernet eth3 address 10.3.34.1/27
vyatta@vyatta2# set interfaces ethernet eth4 address 10.3.34.33/27
vyatta@vyatta2# commit
```

En el router Vyatta 3 se ejecutarán las siguientes instrucciones para la configuración de los interfaces:

```
vyatta@vyatta3:~$ configure
vyatta@vyatta3# set interfaces ethernet eth1 address 10.3.34.129/27
vyatta@vyatta3# set interfaces ethernet eth2 address 10.3.34.161/27
vyatta@vyatta3# set interfaces ethernet eth3 address 10.3.34.65/27
vyatta@vyatta3# set interfaces ethernet eth4 address 10.3.34.97/27
vyatta@vyatta3# commit
```

A continuación se establecerán las rutas estáticas.
En el router Vyatta 1 se configurará una ruta por defecto hacia la dirección 10.3.16.1, y también se añadirá una ruta por cada Red a la que el router no esté conectado directamente:

```
vyatta@vyatta1# set system gateway-address 10.3.16.1

vyatta@vyatta1# set protocols static route 10.3.34.160/27 next-hop
10.3.34.129      → Red F
vyatta@vyatta1# set protocols static route 10.3.34.0/27 next-hop
10.3.34.193      → Red A
vyatta@vyatta1# set protocols static route 10.3.34.32/27 next-hop
10.3.34.193      → Red B
vyatta@vyatta1# set protocols static route 10.3.34.64/27 next-hop
10.3.34.129      → Red C
vyatta@vyatta1# set protocols static route 10.3.34.96/27 next-hop
10.3.34.129      → Red D
```

En el router Vyatta 2 se configurará una ruta por defecto hacia el router Vyatta 1, y también una ruta por cada Red a la que el router no esté conectado directamente:

***vyatta@vyatta2# set system gateway-address*** *10.3.34.194*

***vyatta@vyatta2# set protocols static route*** *10.3.34.128/27* ***next-hop*** *10.3.34.161* → ***Red E***
***vyatta@vyatta2# set protocols static route*** *10.3.34.64/27* ***next-hop*** *10.3.34.161* → ***Red C***
***vyatta@vyatta2# set protocols static route*** *10.3.34.96/27* ***next-hop*** *10.3.34.161* → ***Red D***

En el router Vyatta 3 se configurará una ruta por defecto hacia el router Vyatta 1 y también una ruta por cada Red a la que el router no esté conectado directamente:

***vyatta@vyatta3# set system gateway-address*** *10.3.34.130*

***vyatta@vyatta3# set protocols static route*** *10.3.34.0/27* ***next-hop*** *10.3.34.162* → ***Red A***
***vyatta@vyatta3# set protocols static route*** *10.3.34.32/27* ***next-hop*** *10.3.34.162* → ***Red B***
***vyatta@vyatta3# set protocols static route*** *10.3.34.192/27* ***next-hop*** *10.3.34.162* → ***Red G***

En los PCs habría que configurar una dirección IP según la subred en la que se encuentran y la ruta por defecto.

***usuario@PC1:~$ sudo ifconfig*** *eth0 10.3.34.2* ***netmask*** *255.255.255.224*
***usuario@PC1:~$ sudo route add defautlt gw*** *10.3.34.1*

***usuario@PC2:~$ sudo ifconfig*** *eth0 10.3.34.34* ***netmask*** *255.255.255.224*
***usuario@PC2:~$ sudo route add defautlt gw*** *10.3.34.33*

***usuario@PC3:~$ sudo ifconfig*** *eth0 10.3.34.66* ***netmask*** *255.255.255.224*
***usuario@PC3:~$ sudo route add defautlt gw*** *10.3.34.65*

***usuario@PC4:~$ sudo ifconfig*** *eth0 10.3.34.98* ***netmask*** *255.255.255.224*
***usuario@PC4:~$ sudo route add defautlt gw*** *10.3.34.97*

Después de todas las configuraciones, cada dispositivo ya sería capaz de comunicarse con cualquier otro.
Como se puede observar, el modo de configuración en Vyatta es muy parecido a la de los routers CISCO.

### 3.7.2. Direcciones Secundarias

Se supone la topología de la figura 29. En este caso se empleará un solo interfaz del router, por ejemplo éste podría tener en verdad una sola tarjeta de red. Además, las direcciones de los dos PCs pertenecen a redes diferentes.

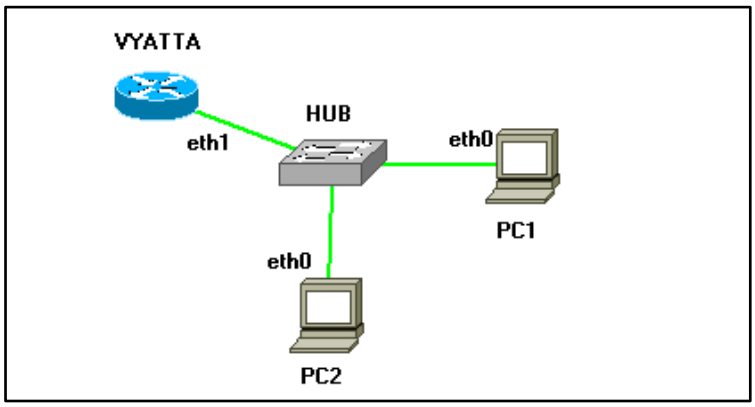


**Figura 29 – Direcciones Secundarias**

Para ello se va a dividir la red 10.3.3.0/24 en dos subredes, 10.3.3.0/25 y 10.3.3.128/25. Se configurarán los PCs, uno en cada red. Dado que cada uno pertenece a una red diferente, no saben cómo llegar al otro. Lo que se va a hacer es configurar en ambos el interfaz del router como router por defecto.

***usuario@PC1:~$ sudo ifconfig*** *eth0 10.3.3.1/25*
***usuario@PC1:~$ sudo route add defautlt gw*** *10.3.3.2* ***netmask*** *255.255.255.128*

***usuario@PC2:~$ sudo ifconfig*** *eth0 10.3.3.129/25*
***usuario@PC2:~$ sudo route add defautlt gw*** *10.3.3.130* ***netmask*** *255.255.255.128*

Sin embargo, la dirección IP del interfaz del router debe pertenecer a la misma red que el interfaz del host, y si ambos PCs están en redes diferentes, eso significa que el interfaz del router debe tener 2 direcciones IP diferentes a la vez. En Vyatta, al igual que en CISCO, esto también es posible.

Para ello solamente se tendrá que configurar a la tarjeta de red Ethernet dos direcciones IP:

***vyatta@vyatta2:~$ configure***
***vyatta@vyatta2# set interfaces ethernet*** *eth1* ***address*** *10.3.3.2/25*
***vyatta@vyatta2# set interfaces ethernet*** *eth1* ***address*** *10.3.3.130/25*

Ahora los PCs se podrán comunicar aunque estén en redes diferentes. Cuando un PC quiera mandar un paquete a otro, éstos tendrán que pasar por el router.

## 3.8. Práctica 8 - Rutas asimétricas. Fragmentación y reensamblado

En esta práctica se modifica el tamaño de la MTU de los interfaces WAN serie de los routers Cisco, y se puede ver cómo dependiendo del tamaño del paquete de datos y del tamaño MTU del interfaz, se producirá la fragmentación del paquete.

El MTU es un parámetro que indica el tamaño máximo que debe tener un datagrama para que sea transmitido por una interfaz sin que necesite ser fragmentado en unidades más pequeñas. El MTU debe ser superior al datagrama más grande que se desea transmitir para que no sea fragmentado. Dicho de otro modo, el MTU expresa el tamaño máximo (en bytes) de un paquete para que pueda ser transmitido de una sola vez.

Como el objetivo de la práctica es configurar los distintos parámetros de velocidad y MTU en los routers para los enlaces serie, al no poder configurar éstos en el sistema Vyatta, por las razones expuestas en la práctica 5, se realizará lo mismo utilizando interfaces Ethernet.

### 3.8.2. Fragmentación en origen

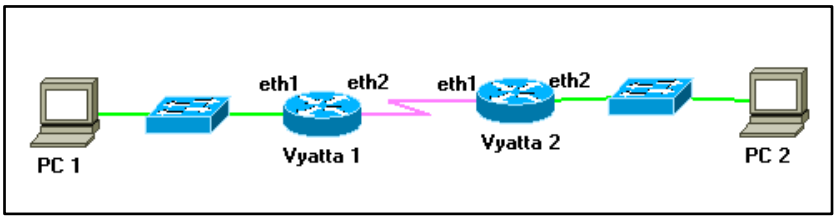


**Figura 30 – Fragmentación en origen**

Para realizar las pruebas, se va a realizar el escenario de la figura 30 utilizando la red 10.3.3.0/24. Para ello se crean 3 subredes (10.3.3.0/26, 10.3.3.64/26 y 10.3.3.128/26) y se procede, al igual que en los demás escenarios, asignando direcciones IP y rutas por defecto. La unión de los routers se hace mediante un cable cruzado Ethernet.

***vyatta@vyatta1# set interfaces ethernet** eth1 **address** 10.3.3.1/26*
***vyatta@vyatta1# set interfaces ethernet** eth2 **address** 10.3.3.65/26*
***vyatta@vyatta1# set system gateway-address** 10.3.3.66*

***vyatta@vyatta2# set interfaces ethernet** eth1 **address** 10.3.3.66/26*
***vyatta@vyatta2# set interfaces ethernet** eth2 **address** 10.3.3.129/26*
***vyatta@vyatta2# set system gateway-address** 10.3.3.65*

***usuario@PC1:~$ sudo ifconfig** eth0 10.3.3.2/26*
***usuario@PC1:~$ sudo route add defautlt gw** 10.3.3.1*

***usuario@PC2:~$ sudo ifconfig** eth0 10.3.3.130/26*
***usuario@PC2:~$ sudo route add defautlt gw** 10.3.3.129*

En primer lugar se va comprobar cuál es el tamaño MTU de una interfaz Ethernet. Para ello se ejecuta en un PC el comando ifconfig, y se ve que es de 1500 bytes.

A continuación, se a comprobar mediante el comando *ping*, el tamaño de datos que envía éste. El tamaño es de 98 bytes. Como la MTU es de 1500 bytes, mayor que el tamaño del paquete ICMP, la trama llega correctamente y sin fragmentar.

En la figura 31 se puede ver el detalle de la trama Ethernet, desencapsulando cada uno de los niveles de la pila de protocolos TCP/IP, indicando el tamaño de los encabezados correspondientes a cada nivel de la pila.

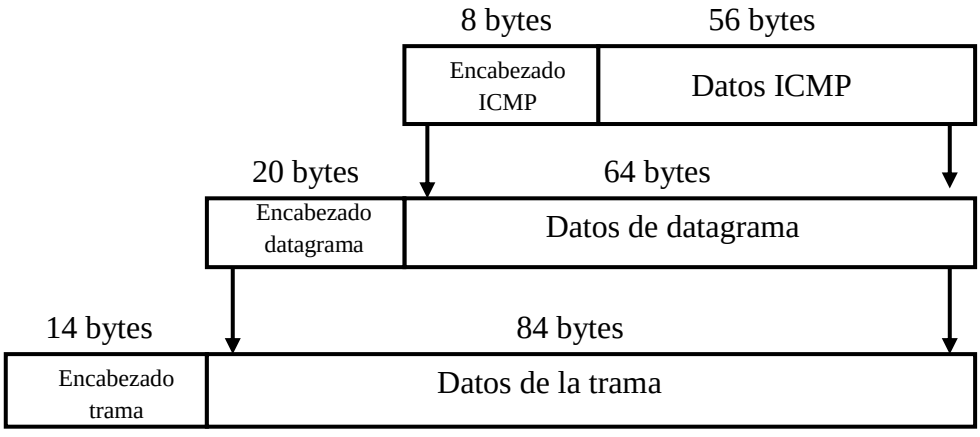


**Figura 31 – Trama Ethernet**

Para probar la fragmentación de las tramas, se va a enviar un solo paquete de 2945 bytes de datos desde PC1 hacia PC2.

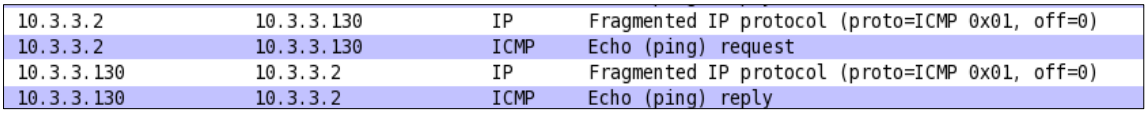


**Figura 32 – Fragmentación trama**

En la figura 32 se aprecia la fragmentación de la trama en dos, ya que el tamaño de la MTU es de 1500 bytes y la trama es mayor. El primero es un paquete IP de 1514 bytes y el segundo ya es el paquete ICMP de 1507 bytes en donde ya se reensambla el primero.

### 3.8.3. Fragmentación en tránsito (ruta simétrica)

Ahora se va a modificar la MTU de los interfaces Ethernet asociados al enlace Ethernet que conecta router1 con router2, de manera que el mayor datagrama que pueda atravesar dicho enlace, sin ser fragmentado, sea como máximo de 1000 bytes.

En primer lugar habrá que deshabilitar los interfaces en los cuales se vaya a modificar el tamaño de la MTU, ya que el sistema Vyatta no permite cambiar este tamaño si los interfaces están activos.

Para ello habrá que ejecutar en cada sistema Vyatta las siguientes instrucciones:

***vyatta@vyatta1# set interfaces ethernet*** *eth2* ***mtu*** *1000*

***vyatta@vyatta2# set interfaces ethernet*** *eth1* ***mtu*** *1000*

Para probar la fragmentación, se envía al igual que antes un paquete de 2549 bytes de datos desde PC1 hacia PC2. En este caso, como el MTU es de 1000 bytes, fragmenta el paquete en 3 de 976 bytes cada uno, y un último ICMP, el cual reensamblará todos los anteriores, formando el paquete de 2945 bytes.

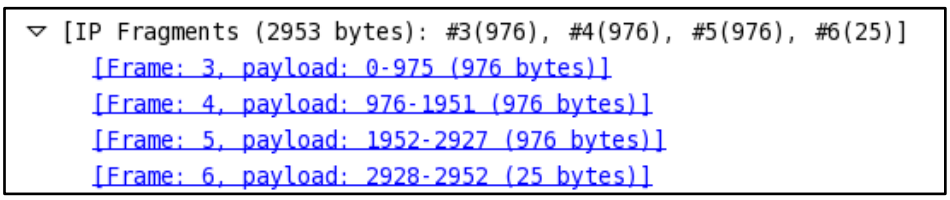


**Figura 33 – Paquete ICMP de reensamblado**

Como se ve, la forma de asignar el valor MTU a las interfaces no es diferente a cómo se haría en Cisco. Lo único que varía es el tipo de interfaz. En los interfaces serie, habría que asignar la velocidad que emplearía. En este escenario, al ser interfaces Ethernet, se le podría asignar la velocidad, pero por defecto es *auto*, lo que significa, que es el sistema el que negocia la velocidad de la interfaz con la interfaz del otro extremo.

## 3.9. Práctica 9 - Configuración de un ISP de acceso por módem

El objetivo de esta práctica es configurar una red en la cual se dé acceso a los usuarios a través de un modem, lo que se conoce como ISP.
Como no intervienen principalmente los routers, no se puede comparar Cisco con Vyatta, por lo que no aporta nada al proyecto.

## 3.10. Práctica 10 - Enrutamiento con RIP

En esta práctica se ve cómo configurar el protocolo de enrutamiento RIP en los routers Cisco.

Para probar que en Vyatta también es posible su configuración y funcionamiento, se configurará el escenario de la figura 34.

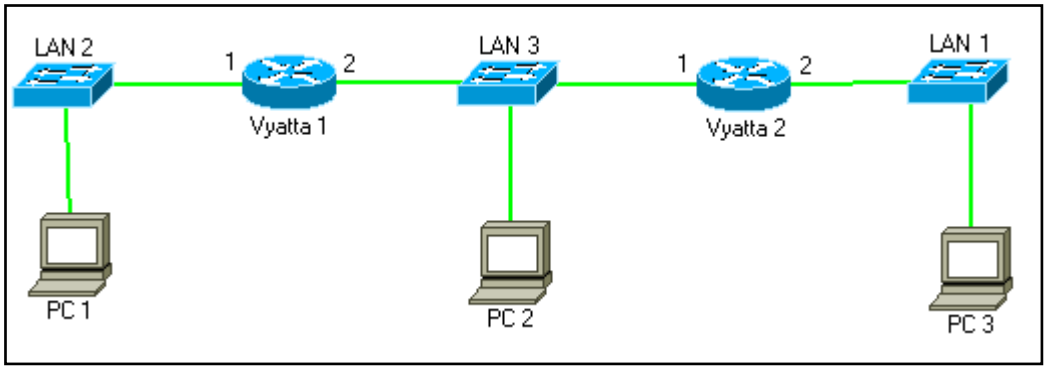


**Figura 34 – Enrutamiento RIP**

Como se puede observar en la figura 34, se dispone de 3 redes:

- LAN 1 → 192.168.1.0/24
- LAN 2 → 192.168.2.0/24
- LAN 3 → 192.168.3.0/24

Se empezará asignando direcciones IP, según la red en la que estén, a los interfaces Ethernet de los routers y los PCs.

```
vyatta@vyatta1:~$ configure
vyatta@vyatta1# set interfaces ethernet eth1 address 192.168.2.1/24
vyatta@vyatta1# set interfaces ethernet eth2 address 192.168.3.1/24
vyatta@vyatta1# commit

vyatta@vyatta2:~$ configure
vyatta@vyatta2# set interfaces ethernet eth1 address 192.168.3.2/24
vyatta@vyatta2# set interfaces ethernet eth2 address 192.168.1.1/24
vyatta@vyatta2# commit

usuario@PC1:~$ sudo ifconfig eth0 192.168.2.2 netmask 255.255.255.0
usuario@PC1:~$ sudo route add defautlt gw 192.168.2.1

usuario@PC2:~$ sudo ifconfig eth0 192.168.3.3 netmask 255.255.255.0
usuario@PC2:~$ sudo route add defautlt gw 192.168.3.1

usuario@PC3:~$ sudo ifconfig eth0 192.168.1.2 netmask 255.255.255.0
usuario@PC3:~$ sudo route add defautlt gw 192.169.1.1
```

Ahora si se intenta enviar un paquete desde el PC1 al PC3 se ve que no es posible, debido a que los routers no saben qué camino deben tomar los paquetes que van a la redes a las que no están conectados, ya que no tienen establecido ningún tipo de ruta estática.

Para ello se va a configurar el protocolo RIP en cada uno de los routers. Su configuración es muy sencilla.

Como primer paso se va a desactivar el mecanismo llamado *split horizon*, que es el que evita las cuentas a infinito en ciertas situaciones. Se puede desactivar, al igual que en CISCO, en cada interfaz de forma independiente.

Para ello se va a cada router y se ejecuta en modo configuración:

```
vyatta@vyatta1# set interfaces ethernet eth1 ip rip split-horizon disable
vyatta@vyatta1# set interfaces ethernet eth2 ip rip split-horizon disable

vyatta@vyatta2# set interfaces ethernet eth1 ip rip split-horizon disable
vyatta@vyatta2# set interfaces ethernet eth2 ip rip split-horizon disable
```

A continuación se activa el proceso RIP. A diferencia de CISCO, en Vyatta no hay que ejecutar una instrucción para activar el proceso, y luego otra para decirle en que redes se quiere que emplee RIP.

```
vyatta@vyatta1# set protocols rip network 192.168.3.0/24
vyatta@vyatta1# set protocols rip redistribute connected

vyatta@vyatta2# set protocols rip network 192.168.3.0/24
vyatta@vyatta2# set protocols rip redistribute connected
```

Solamente se ejecutará una instrucción para anunciar las redes que están conectadas a otro router y otra instrucción que será la que anuncie las redes directamente conectadas.

Y con esto ya estaría configurado y listo para poder acceder a todas las partes de la red.

| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 1 | 0.000000 | 192.168.3.1 | 224.0.0.9 | RIPv2 | Response |
| 2 | 3.855625 | 192.168.3.2 | 224.0.0.9 | RIPv2 | Response |
| 3 | 32.869346 | 192.168.3.2 | 224.0.0.9 | RIPv2 | Response |
| 4 | 34.004044 | 192.168.3.1 | 224.0.0.9 | RIPv2 | Response |

**Figura 35 – Captura paquete RIP**

En la figura 35 se puede ver cómo cada interfaz de red de cada router, envía las rutas para que los otros routers las aprendan.
También se puede apreciar en la imagen que envía las rutas cada 30 segundos.

Se puede cambiar el tiempo en el que el router anuncia sus rutas, mediante los paquetes RIP, con la siguiente instrucción:

```
vyatta@vyatta# set protocols rip timers update seconds
```

Con la instrucción ***show ip route*** se puede comprobar si ha aprendido bien las rutas y ver a qué redes está conectado directamente y mediante qué interfaz.

```
vyatta@vyatta1:~$ show ip route

Codes: K – kernel route, C – connected, S – static , R – RIP, O – OSPF,
       I – ISIS, B – BGP, > - selected route, * - FIB route.

C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.1.0 is directly connected, eth1
C>* 192.168.2.0 is directly connected, eth2
R>* 192.168.3.0 [120/2] via 192.168.3.2, eth2, 00:02:14
```

Para verlo de una manera más detallada, en la que muestre las redes conectadas directamente al router, las que ha aprendido y cuál es su salto, etc, se usará la siguiente instrucción:

***vyatta@vyatta1:~$ show ip rip***

*Codes: K – kernel route, C – connected, S – static , R – RIP, O – OSPF, B – BGP*
*Sub-codes:*
*(n) - normal, (s) - static, (d) - default, (r) - redistribute, (i) - interface*

| | Network | Next Hop | Metric | From | Tag | Time |
|---|---|---|---|---|---|---|
| C(n) | 192.168.2.0/24 | 0.0.0.0 | 1 | self (connected:1) | 0 | |
| C(n) | 192.168.3.0/24 | 0.0.0.0 | 1 | self | 0 | |
| R(n) | 192.168.1.0/24 | 192.168.3.2 | 2 | 192.168.3.2 | 0 | 2:55 |

Se observa que para ir a la red 192.168.1.0, que no está conectado directamente, el paquete tiene que dar un salto a la IP 192.168.3.2, que es la interfaz eth1 del router Vyatta 2, y a través de él acceder a la red 1.

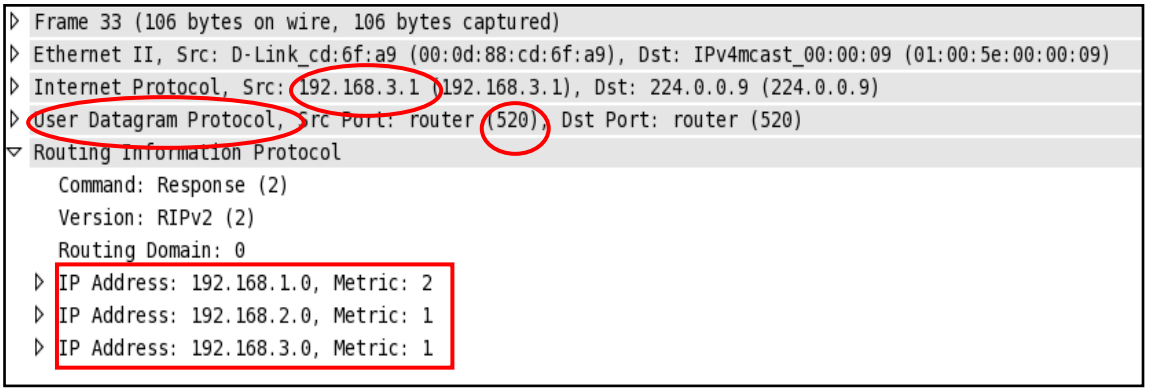


**Figura 36 - Detalle paquete RIP**

En la figura 36 se puede ver los detalles del paquete que envía el interfaz de red del router 1 (192.168.3.1). El protocolo que utiliza es UDP y el puerto al que van dirigidos es al 520.
Y lo más importante, se puede ver la tabla de rutas que envía. En este caso, se ve que informa a los otros routers que está conectado directamente a las redes 192.168.2.0 y 192.168.3.0, y que para ir a la red 192.168.1.0, tiene que dar 2 saltos (se puede comprobar por dónde van con los comandos dichos anteriormente).

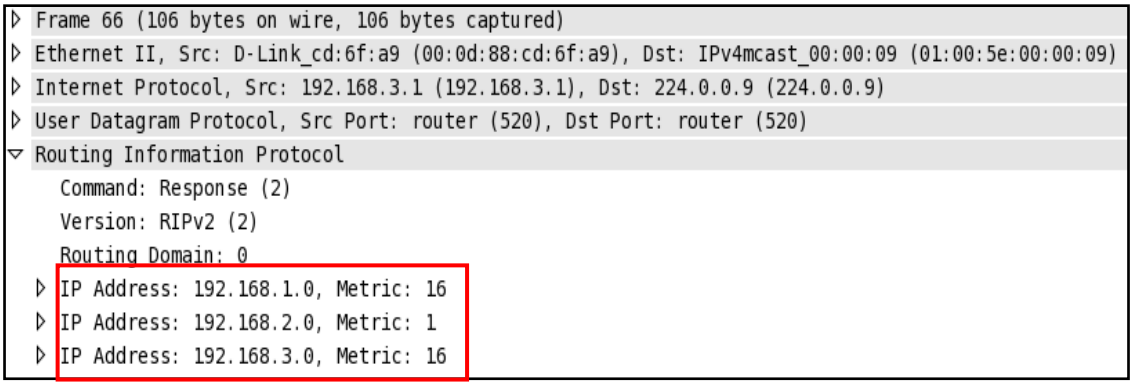


**Figura 37 – Desconexión interfaz**

Si por un casual se desconectara una interfaz de algún router, esa red ya no sería accesible por otros routers, y por lo tanto, las tablas de rutas se verían modificadas.
Para comprobar esto, se va a desconectar la interfaz eth2 del router Vyatta1. Al hacerlo, el router desactiva automáticamente su interfaz. El router vyatta2, no sabe que vyatta1 se ha desconectado de la red, y sigue pensando que se puede llegar a la LAN2 a través de él. Transcurrido cierto tiempo, en concreto 180 segundos, vyatta2 marca esa red como posiblemente inalcanzable y empieza a mandar los paquetes de RIP con métrica 16 en esa ruta.

Se puede cambiar el tiempo que tarda el router en actualizar su tabla de rutas mediante:

> ***vyatta@vyatta# set protocols rip timers timeout*** *seconds*

Este mecanismo de enviar la ruta inválida con métrica 16 en vez de simplemente borrarla, se conoce como *poison reverse*, y reduce el tiempo de convergencia, es decir, el tiempo que tardan los routers de la red en enterarse de cambios en la misma.

En Vyatta se puede habilitar mediante la siguiente instrucción:

> ***vyatta@vyatta# set interfaces*** *interface* ***ip rip split-horizon poison-reverse***

En la figura 37 se ve cómo el router Vyattta1 anuncia que no puede ir a las redes 192.168.1.0 y 192.168.3.0 debido a que se ha desconectado la interfaz eth1, y por lo tanto solo está conectado a la red 192.168.2.0.

## 3.11. Práctica 11 - Enrutamiento con RIP en un escenario heterogéneo de equipos Cisco y Linux

El objetivo de este apartado es ver cómo puede funcionar todo correctamente al combinar distintos tipos de routers. En este caso se comprobará la interacción del sistema Vyatta con un Linux. A simple vista no debe suponer ningún problema, ya que no hay que olvidarse que el sistema Vyatta está basado en una distribución de Linux.

El programa típico en Unix se llama *routed*, pero en este caso se va a utilizar uno más flexible que se llama *gated*. Para ello se creará un archivo llamado *gated.conf* que será un fichero de configuración que gated cargará cuando lo ejecute.

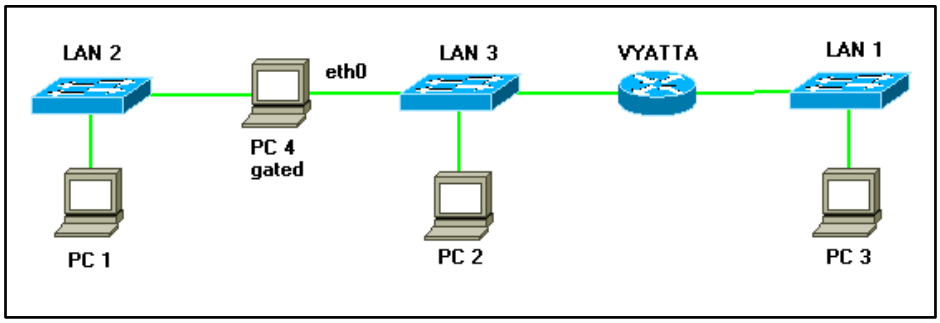


**Figura 38 – Vyatta con gated.**

Se va a habilitar el PC 4 de la figura 38, como un router en el que corre un proceso RIP. Para ello habrá que configurar el archivo gated.conf de la siguiente manera:

```
rip yes {
        interface eth0 version 2 multicast;
};
```

Con estas instrucciones se configurará el protocolo RIP, en los dos interfaces Ethernet, en la versión 2, ya que el sistema Vyatta admite la versión 1, pero no se puede elegir en qué momento usar una u otra.

Después se arrancará el servicio gated con:

```
usuario@PC1:~$ sudo gated –N –t –f gated.conf
```

Se dispone de 3 redes, LAN 1 (192.168.1.0/24), LAN 2 (192.168.2.0) y LAN 3 (192.168.3.0/24).
Se asignan las direcciones a los interfaces de los PCs y las rutas por defecto.

***usuario@PC1:~$ sudo ifconfig** eth0 192.168.2.1 **netmask** 255.255.255.0*
***usuario@PC1:~$ sudo route add defautlt gw** 192.168.2.2*

***usuario@PC2:~$ sudo ifconfig** eth0 192.168.3.3 **netmask** 255.255.255.0*
***usuario@PC2:~$ sudo route add defautlt gw** 192.168.3.1*

***usuario@PC3:~$ sudo ifconfig** eth0 192.168.1.2 **netmask** 255.255.255.0*
***usuario@PC3:~$ sudo route add defautlt gw** 192.168.1.1*

***gated@PC4:~$ sudo ifconfig** eth1 192.168.2.2 **netmask** 255.255.255.0*
***gated@PC4:~$ sudo ifconfig** eth0 192.168.3.1 **netmask** 255.255.255.0*

Se habilita el reenvío (forwarding) de paquetes en el PC que actuará como router.
Por último se activa el proceso RIP en el PC, y en el sistema Vyatta como ya se vio anteriormente.

A partir de ahí, los dos routers se intercambian las redes sin ningún problema.
En la figura 39 se puede ver las rutas que ha aprendido el PC 4.

```
[rba@pcB ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
224.0.0.9       localhost.local 255.255.255.255 UGH   0      0        0 lo
localhost.local localhost.local 255.255.255.255 UGH   0      0        0 lo
192.168.3.0     192.168.3.1     255.255.255.0   UG    0      0        0 eth0
192.168.3.0     *               255.255.255.0   U     0      0        0 eth0
192.168.2.0     192.168.2.2     255.255.255.0   UG    0      0        0 eth1
192.168.2.0     *               255.255.255.0   U     0      0        0 eth1
192.168.1.0     192.168.3.2     255.255.255.0   UG    0      0        0 eth0
loopback        -               255.0.0.0       !     0      -        0 -
```

**Figura 39 – Route gated**

Se observa que ha aprendido que para ir la LAN 1 (192.168.1.0/24), tiene que ir por la interfaz eth0 y por la dirección 192.168.3.2.
Para finalizar se comprueba la conexión entre los extremos y todo parece funcionar correctamente.

## 3.12. Práctica 12 - Network Address Translation (NAT)

En esta práctica se va a ver cómo una red puede hacer parecer al exterior que emplea un espacio de direcciones diferentes del que usa en realidad. Para ello se van a probar dos métodos, la conversión estática y la dinámica.

La conversión de direcciones de red o NAT se desarrolló para resolver la falta de direcciones IP con el protocolo IPv4. El principio de NAT consiste en utilizar una conexión de pasarela a Internet, que tenga al menos una interfaz de red conectada a la red interna y al menos una interfaz de red conectada a Internet (con una dirección IP enrutable) para poder conectar todos los equipos a la red. Cuando un paquete está saliendo del dominio, NAT convierte la dirección IP origen en una dirección pública. Y cuando un paquete entra en el dominio, NAT convierte la dirección pública destino en la dirección local apropiada.

### 3.12.1. Conversión estática

Para probar todo esto, se creará el escenario de la figura 40 en el que se hará una conversión NAT de uno a uno.

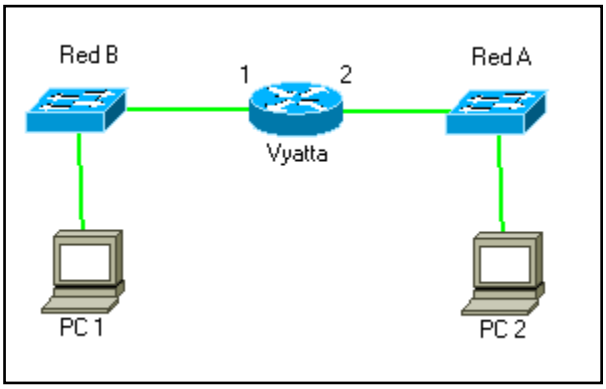


**Figura 40 – NAT**

La Red B representará el interior y la Red A el exterior. En el interior se empleará el espacio de direcciones 192.168.0.0/24. Y para el exterior, el espacio de direcciones 10.0.0.0/16.

Al igual que todos los servicios de Vyatta primero se activará el servicio con la instrucción ***set service nat***. No hace falta hacer esto por separado, ya que al escribir luego las reglas NAT habrá que escribirlo cada vez. La configuración final quedaría así:

```
vyatta@vyatta# set service nat rule 10 type source
vyatta@vyatta# set service nat rule 10 source address 192.168.0.2
vyatta@vyatta# set service nat rule 10 outbound-interface eth2
vyatta@vyatta# set service nat rule 10 outside-address address 10.0.0.1
```

En el sistema Vyatta, NAT es configurado mediante una serie de reglas. Cada regla sirve para llevar a cabo una serie de traducciones de direcciones de red. Estas reglas están numeradas, y son evaluadas en orden numérico. Una vez configurado el número de regla, éste ya no se puede cambiar, por lo que si se quiere modificar la regla, habrá

que borrarla y crearla otra vez con el número que se quiera. Por ésta razón, es buena práctica crear reglas dejando un espacio considerable (de unos 10 números).
Como se ve, se ha creado la regla número 10, dejando espacio anteriormente por si se tiene la necesidad de crear una regla nueva más restrictiva, ya que una vez creada la regla no se podrá modificar su número.

Se ha establecido que sea de tipo *source* (origen) ya que se va a traducir la dirección de red origen (SNAT), que en este caso solo se va a tratar de una IP. Se define también la interfaz que va a transmitir el tráfico de salida, en este caso eth2. Y por último se establece la dirección por la que se traducirá.

Si se quisiera traducir toda la red, tan solo habría que cambiar en la instrucción "***set service nat rule 10 source address 192.168.0.2"*** la dirección IP por ***192.168.0.0/24***.
Para ver la regla NAT que se ha creado se utiliza la instrucción:

```
vyatta@vyatta# show service nat rule 10
  rule 10 {
     outbound-interface eth2
     outside-address  {
       address  10.0.0.1
     }
     source  {
        address 192.168.0.2
     }
     type source
  }
```

Por último, se podría ver las traducciones que va haciendo el sistema con la siguiente instrucción:

```
vyatta@vyatta~$ show nat translations
Pre-NAT         Post-NAT        Type        Prot        Timeout
192.168.0.2     10.0.0.1        SNAT        ICMP        29
```

A diferencia de Cisco, en Vyatta no hay que definir qué interfaz está conectado al exterior y cuál al interior, sino que solo habrá que definir la interfaz que va a transmitir el tráfico de salida, la cual será la encargada de realizar la conversión de direcciones.
Lo que sí se mantiene es que se tendrá que definir una regla para la conversión estática de las direcciones.

### 3.12.2. Conversión Dinámica

A continuación se creará un conjunto (pool) de direcciones que el router empleará para convertir las direcciones internas a ellas.

Manteniendo el escenario de la figura 40, se procedería con las siguientes instrucciones:

```
vyatta@vyatta# set service nat rule 10 type source
vyatta@vyatta# set service nat rule 10 source address 192.168.0.0/24
vyatta@vyatta# set service nat rule 10 outbound-interface eth2
vyatta@vyatta# set service nat rule 10 outside-address address 10.0.0.1-
10.0.0.4
```

Como se observa, los pasos serían los mismos que para una conversión estática, con la única variación de la última regla, en la que se define el rango de direcciones por las que se podrá convertir la dirección IP.

En Cisco, habría que definir el rango de direcciones públicas, asignándole un nombre y crear una lista de acceso con las direcciones de las máquinas a las que se convertirán. Y por último indicarle que esa lista de acceso contiene las IPs que se quieren convertir.

### 3.12.3. Sobrecarga de la dirección exterior del router

En este apartado se va a configurar el router para que convierta todas las direcciones internas en su dirección externa.

Existe una opción denominada *masquerade,* la cual es utilizada en escenarios en los cuales, los dispositivos de una red privada tienen que acceder al exterior, normalmente Internet, y han de convertir su dirección privada en una sola dirección pública. El escenario de la figura 40, podría ser configurado también mediante esta opción.

```
vyatta@vyatta# set service nat rule 10 type masquerade
vyatta@vyatta# set service nat rule 10 source address 192.168.0.0/24
vyatta@vyatta# set service nat rule 10 outbound-interface eth2
```

Con este método no haría falta establecer la dirección pública por la que traducir las privadas, ya que enmascararía todas por la dirección de la interfaz eth2.A diferencia de Cisco, en Vyatta no se tendrá que crear listas de acceso.
Una vez configurado, se conecta otro PC en la red B, y se conectan los dos simultáneamente al PC2 de la red A. Como resultado, se ve que los dos PCs de la red B tienen la misma dirección, la cual ha sido traducida por el servicio NAT del router.

# 4.- CONCLUSIÓN

Cisco ha diseñado, y sigue diseñando, un sistema operativo muy complejo, el cual está por encima de las necesidades de la mayoría de los usuarios. Su dependencia de los procesadores especializados y hardware, han disminuido la capacidad de Cisco para innovar. Por otro lado, su éxito es evidente, debido al gran número de usuarios y de empresas que utilizan estos routers, a pesar de su alto precio. También decir que las pruebas realizadas en el proyecto son a un nivel de laboratorio, por lo que los resultados no se pueden comparar con los que resultarían en una gran empresa en dónde un router tiene que manejar millones de paquetes por segundo con docenas de miles de rutas.

El sistema de código abierto de Vyatta, en cambio utiliza software libre y el sistema se puede instalar en PCs con arquitectura x86. Vyatta ha creado una solución de protocolos de enrutamiento y seguridad, asequible y comparable a Cisco. Al eliminar la dependencia de hardware especializado, Vyatta abre nuevas posibilidades en el ámbito de las tecnologías de redes.

Después de haber realizado las prácticas de LPR, se puede comprobar que a través del sistema Vyatta es posible realizar todas las configuraciones que antes se hacían en Cisco, y por lo tanto realizar los mismos escenarios. Además, incluye distintos servicios para garantizar la seguridad de la red, y por lo tanto permite tener todo unificado. En ocasiones los comandos de configuración son más sencillos en el sistema Vyatta, aunque en general, el método de configurar todos los parámetros es muy parecido.

Por otro lado, ha habido algunos aspectos que Vyatta ha decidido que para poder configurarlos, y por lo tanto utilizarlos, hay que pagar por una versión que los incluye y da soporte. Otro aspecto que no funcionó como se esperaba, fue la virtualización del sistema, el cual dio problemas al reconocer los interfaces Ethernet, por lo que se tuvo que tomar la decisión de realizar todas las pruebas mediante la opción de Live USB.
Además el sistema está en constante desarrollo, hace unos días el equipo de Vyatta anunció la nueva versión (VC 6.3) y cada vez el número de usuarios va incrementando, exponiendo sus dudas en foros y páginas web, por lo que los problemas que puedan surgir se van solucionando con la ayuda de todos.

# BIBLIOGRAFÍA


[1] www.xorp.org

[2] www.zebra.org

[3] www.quagga.net

[4] http://bird.network.cz

[5] www.vyatta.com

[6] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_Services_R6.3_v01.pdf

[7] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_NAT_R6.3_v01.pdf

[8] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_RoutingPolicies_R6.3_v01.pdf

[9] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_RIP_R6.3_v01.pdf

[10] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_BGP_R6.3_v01.pdf

[11] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_OSPF_R6.3_v01.pdf

[12] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_Firewall_R6.3_v01.pdf

[13] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_Security_R6.3_v01.pdf

[14] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_VPN_R6.3_v01.pdf

[15] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_QoS_R6.3_v01.pdf

[16] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_RemoteAccessAPI2.0_R6.3_v01.pdf

[17] http://www.vyatta.com/downloads/documentation/VC6.3/Vyatta_WANInterfaces_R6.3_v01.pdf